Dhirubhai Ambani Institute of Information and Communication Technology

IT214 – Database Management System

Database to manage online competitive Coding Platform

Lab Group - 5 : Team - 2 (G5-T2)

Group Members :-

Hriday Asnani - 202301282

Karthik Raman Balamurugan - 202301407

Alongside in separate textfiles are:

- DDL Script
- INSERT statements' script
- Queries' SQL Statements

Special Note : Our project only deals with the data storing and retrieval aspects (database management) of the online coding platform, i.e., the area concerned with our course. However, we have not looked into the running and evaluation of codes, since it is outside the scope of DBMS.

We are considering that our project requires and depends on an external compiler/IDE to compile codes, run them, compare them and evaluate them, in order to make it an actual complete competitive coding platform.

Our project is treating all these aspects (program codes and others) as simple data provided by the compiler software.

Nevertheless, we can assure that we have dealt with the portion concerning DBMS (our course) to the best of our knowledge and hard work to get as close as we can to an actual platform.

Dhirubhai Ambani Institute of Information and Communication Technology
DA-IICT

(1) Minimal FD Set

userid -> Name
userid -> email
userid -> phone_no
userid -> rating
userid -> pwd
userid -> reg_date
userid -> country
probid -> statement
probid -> max_memory_mb
probid -> max_runtime_s
probid -> difficulty
probid -> prob_name
probid ->creator_id
probid -> contestid
{probid, testcaseno} -> input
discid -> topic
postid ->comment
postid -> posted_at
postid -> userid
postid -> discid
contestid -> contest_name
constestid -> duration_hrs
constestid -> contest_ div
contestid -> contest_date_time
{userid , contestid } -> rank
{userid , contestid } -> score
{userid , contestid } -> time_spent_hrs
subid -> submission_time
subid -> code
subid -> memory_s
subid -> status

subid -> language
subid -> runtime_s
subid -> userid
subid ->probid
subid -> wrong_testcaseno
solutionid -> language
solutionid -> code
solutionid -> sol_memory_mb
solutionid -> sol_runtime_s
solutionid -> userid
solutionid -> probid

(2) Proof that relations are in BCNF

1. 'Users' relation :

- Attributes :

Users {Userid, Name, email, phone_no, rating, pwd, reg_date, country}

- Functional dependencies :

userid -> Name
userid -> email
userid -> phone_no
userid -> rating
userid -> pwd
userid -> reg_date
userid -> country

Let X = User_id
X^+ = {userid, Name, email, phone_no, rating, pwd, reg_date, country}
Thus, **Primary key = userid**

We can see that the left side of all the FDs in the minimal set of FDs for the relation 'Users' is userid, which is the primary key of this relation, so "Users" **is in BCNF.**

2. 'Friend_of' relation:

- Attributes:

  Friend_of {userid, friendid}

- The only functional dependency existing here is the trivial one:

  {userid, friendid} -> {userid, friendid}

  Thus, the entire set of attributes, i.e, **{userid, friendid}** is the primary key and the relation complies with BCNF as it is not violating any of its rules.
  We can see that the left side of the only FD in the minimal FD set for this relation is equivalent to the Primary Key itself, so "Friend_of" **is in BCNF.**

3. ‘Problems’ relation :

- Attributes :

Problems {probid, statement, memory, max_runtime, difficulty, prob_name, tags,creator_id,contestid}

- Functional dependencies :

probid -> prob_name
probid -> statement
probid -> max_memory_mb
probid -> max_runtime_s
probid -> difficulty
probid -> tags
probid -> creator_id
probid -> contestid

Let X = probid
X^+ = {probid, statement,max_ memory_mb, max_runtime_s, difficulty, prob_name, tags, creator_id, contestid}
Thus, **Primary key = probid**

We can see that the left side of all the FDs in the minimal set of FDs for the relation ‘Problems’ is probid , which is the primary key of this relation, so “Problems” **is in BCNF.**

4. ‘Tags’ Relation:

- Attributes:

  Friend_of {probid, tagname}

- The only functional dependency existing here is the trivial one:

  {probid, tagname} -> {probid, tagname}

  Thus, the entire set of attributes, i.e, **{probid, tagname}** is the primary key and the relation complies with BCNF as it is not violating any of its rules.

  We can see that the left side of the only FD in the minimal FD set for this relation is equivalent to the Primary Key itself, so “Tags” **is in BCNF.**

5. ‘Favorites’ Relation:

- Attributes:

  Friend_of {userid, probid}
- The only functional dependency existing here is the trivial one:

  {userid, probid} -> {userid, probid}

  Thus, the entire set of attributes, i.e, **{userid, probid}** is the primary key and the relation complies with BCNF as it is not violating any of its rules.

  We can see that the left side of the only FD in the minimal FD set for this relation is equivalent to the Primary Key itself, so “Favorites” **is in BCNF.**

6. ‘Test_cases’ relation :

- Attributes :

Test_cases {probid, testcaseno, input}

- Functional dependencies :

  {probid, testcaseno} -> input

Let X = {probid, testcaseno}
X^+ = {probid, testcaseno, input}
Thus, **Primary key = {probid, testcaseno}**

We can see that the left side of all the FDs in the minimal set of FDs for the relation ‘Test_cases’ is {probid, testcaseno} , which is the primary key of this relation, so “Test_cases” **is in BCNF.**

7. ‘Output’ Relation:

- Attributes:

  Friend_of { probid, testcaseno, output}
- The only functional dependency existing here is the trivial one:

  {probid, testcaseno, output} -> {probid, testcaseno, output}

  Thus, the entire set of attributes, i.e, **{probid, testcaseno, output}** is the primary key and the relation complies with BCNF as it is not violating any of its rules.
  We can see that the left side of the only FD in the minimal FD set for this relation is equivalent to the Primary Key itself, so “Output” **is in BCNF.**

8. ‘Discussions’ relation :

- Attributes :

Discussions {discid, topic}

- Functional dependencies :

  discid -> topic

Let X = discid
X^+ = {discid, topic}
Thus, **Primary key = discid**

We can see that the left side of all the FDs in the minimal set of FDs for the relation Discussions is discid , which is the primary key of this relation, so “Discussions” **is in BCNF.**

9. ‘About’ Relation:

- Attributes:

  Friend_of { discid,probid}
- The only functional dependency existing here is the trivial one:

  {discid,probid} -> {discid,probid}

  Thus, the entire set of attributes, i.e, **{discid, probid}** is the primary key and the relation complies with BCNF as it is not violating any of its rules.

  We can see that the left side of the only FD in the minimal FD set for this relation is equivalent to the Primary Key itself, so “About” **is in BCNF.**

10. ‘Posts’ Relation:

- Attributes :

Posts {postid, comment, posted_at, userid, discid}

- Functional dependencies :

  postid -> comment
  postid -> posted_at
  postid -> userid
  postid -> discid

Let X = postid
X^+ = {postid, comment, posted_at, userid, discid}
Thus, **Primary key = postid**

We can see that the left side of all the FDs in the minimal set of FDs for the relation ‘Posts’ is postid , which is the primary key of this relation, so “Posts” **is in BCNF.**

11. ‘Contests’ relation :

- Attributes :

Contests {contestid, contest_name, contest_div, duration, contest_date_time}

- Functional dependencies :

contestid -> contest_name
contestid -> contest_div
contestid -> duration_hrs
contestid -> contest_date_time

Let X = contestid
X^+ = {contest_name, contest_div, duration_hrs, contest_date_time}
Thus, **Primary key = contestid**

We can see that the left side of all the FDs in the minimal set of FDs for the relation ‘Contests’ is contestid , which is the primary key of this relation, so “Contests” **is in BCNF.**

12. ‘Participates_in’ relation :

- Attributes :

Participates_in {userid, contestid, rank, score, time_spent_hrs}

- Functional dependencies :

{userid, contestid} -> rank
{userid, contestid} -> score
{userid, contestid} -> time_spent_hrs

Let X = {userid, contestid}
X^+ = {userid, contest_id, rank, score, time_spent_hrs}
Thus, **Primary key = {userid, contestid}**

The left side of all the FDs in the minimal set of FDs for the relation ‘Participates_in’ is {userid, contestid}, which is the primary key of this relation, so **“**Participates_in**” is in BCNF.**

13. ‘Written_by’ Relation:

- Attributes:

  Friend_of { userid,contestid}

- The only functional dependency existing here is the trivial one:

  {userid,contestid} -> {userid, contestid}

  Thus, the entire set of attributes, i.e, **{userid, contestid}** is the primary key and the relation complies with BCNF as it is not violating any of its rules.

  We can see that the left side of the only FD in the minimal FD set for this relation is equivalent to the Primary Key itself, so “Written_by” **is in BCNF.**

14. ‘Submissions’ relation :

- Attributes :
  Submissions {subid, code, memory_mb, submission_time, status, language, runtime_s,wrong_testcaseno, userid, probid}
- Functional dependencies :

  subid -> submission_time
  subid -> code
  subid -> memory_mb
  subid -> status
  subid ->language
  subid -> runtime_s
  subid -> userid
  subid -> probid
  subid -> wrong_testcaseno

Let X = subid

X^+ = {subid, code, memory_mb, submission_time, status, language, runtime_s, wrong_testcaseno ,userid, probid}

Thus, **Primary key = subid**

We can see that the left side of all the FDs in the minimal set of FDs for the relation ‘Submissions’ is **subid** , which is the primary key of this relation, so “Submissions” **is in BCNF.**

15. ‘Solution’ relation :

- Attributes :
  Solutions {solutionid, code, language, sol_memory_mb, sol_runtime_s , userid, probid}

- Functional dependencies :

  solutionid -> language
  solutionid -> code
  solutionid -> sol_memory_mb
  solutionid -> sol_runtime_s
  solutionid -> userid
  solutionid -> probid

Let X = solutionid
X^+ = {solutionid, code, language, sol_memory_mb, sol_runtime_s , userid, probid}
Thus, **Primary key = solutionid**

The left side of all the FDs in the minimal set of FDs for the relation ‘Solutions’ is **solutionid**, which is the primary key of this relation, so **“**Solutions**” is in BCNF.**

- **List of Top 5 Queries for the Coding Platform database : -**

1. **Profile of all users including number of problems solved, contests participated, friends, favorite problems along with avg runtime and memory of submissions and most frequently used programming language:**

Sql query :

```
SELECT
    u.userid,
    u.name,
    u.email,
    u.phone_no,
    u.rating,
    u.reg_date,
    u.country,

    solved.count_solved AS problems_solved,
    contests.count_contests AS contests_participated,
    friends.count_friends AS number_of_friends,
    favs.count_favorites AS favorite_problems,
    avg_stats.avg_runtime AS avg_runtime_s,
    avg_stats.avg_memory AS avg_memory_mb,
    lang_stats.most_used_language

FROM CodingPlatform.users u

-- Problems Solved
LEFT JOIN (
    SELECT userid, COUNT(DISTINCT probid) AS count_solved
    FROM CodingPlatform.submissions
    WHERE status = 'Accepted'
    GROUP BY userid
) solved ON u.userid = solved.userid
```

```
-- Contests Participated
LEFT JOIN (
   SELECT userid, COUNT(*) AS count_contests
   FROM CodingPlatform.participates_in
   GROUP BY userid
) contests ON u.userid = contests.userid

-- Friends Count
LEFT JOIN (
   SELECT userid, COUNT(*) AS count_friends
   FROM CodingPlatform.friend_of
   GROUP BY userid
) friends ON u.userid = friends.userid

-- Favorite Problems
LEFT JOIN (
   SELECT userid, COUNT(*) AS count_favorites
   FROM CodingPlatform.favorites
   GROUP BY userid
) favs ON u.userid = favs.userid

-- Average Runtime and Memory
LEFT JOIN (
   SELECT userid,
        AVG(runtime_s) AS avg_runtime,
        AVG(memory_mb) AS avg_memory
   FROM CodingPlatform.submissions
   GROUP BY userid
) avg_stats ON u.userid = avg_stats.userid

-- Most Frequently Used Language
LEFT JOIN (
   SELECT userid, language AS most_used_language
   FROM (
      SELECT userid, language,
```

```
        RANK() OVER (PARTITION BY userid ORDER BY COUNT(*) DESC) AS lang_rank
    FROM CodingPlatform.submissions
    GROUP BY userid, language
  ) ranked_langs
  WHERE lang_rank = 1
) lang_stats ON u.userid = lang_stats.userid;
```

Output :

	userid integer	name character varying (18)	email character varying (30)	phone_no character varying (16)	rating integer	reg_date timestamp without time zone	country character varying (20)	problems_solved bigint	contests_participated bigint	number_of_friend bigint
1	1	Alice	alice@example.com	1234567890	1500	2024-12-08 00:00:00	USA	1	2	
2	2	Bob	bob@example.com	2345678901	1300	2024-11-15 00:00:00	Canada	[null]	2	
3	3	Charlie	charlie@example.com	3456789012	1200	2025-01-05 00:00:00	UK	[null]	2	
4	4	Diana	diana@example.com	4567890123	1600	2023-07-12 00:00:00	USA	1	2	
5	5	Evan	evan@example.com	5678901234	1100	2024-03-20 00:00:00	Australia	2	2	
6	6	Fiona	fiona@example.com	6789012345	1400	2024-10-31 00:00:00	Germany	1	2	
7	7	Gautam	gautam@sample.in	9012345678	1550	2024-08-09 00:00:00	India	2	2	
8	8	Harsha	harsha@sample.in	9123456780	1250	2024-09-30 00:00:00	India	2	2	
9	8	Harsha	harsha@sample.in	9123456780	1250	2024-09-30 00:00:00	India	2	2	
10	9	Isha	isha@sample.in	9234567801	1350	2024-12-25 00:00:00	India	[null]	2	
11	10	Jai	jai@sample.in	9345678012	1500	2025-02-14 00:00:00	India	1	2	
12	11	Kiran	kiran@sample.com	9456780123	1450	2023-11-03 00:00:00	USA	1	2	
13	12	Lalit	lalit@sample.in	9567801234	1600	2025-03-01 00:00:00	India	1	2	

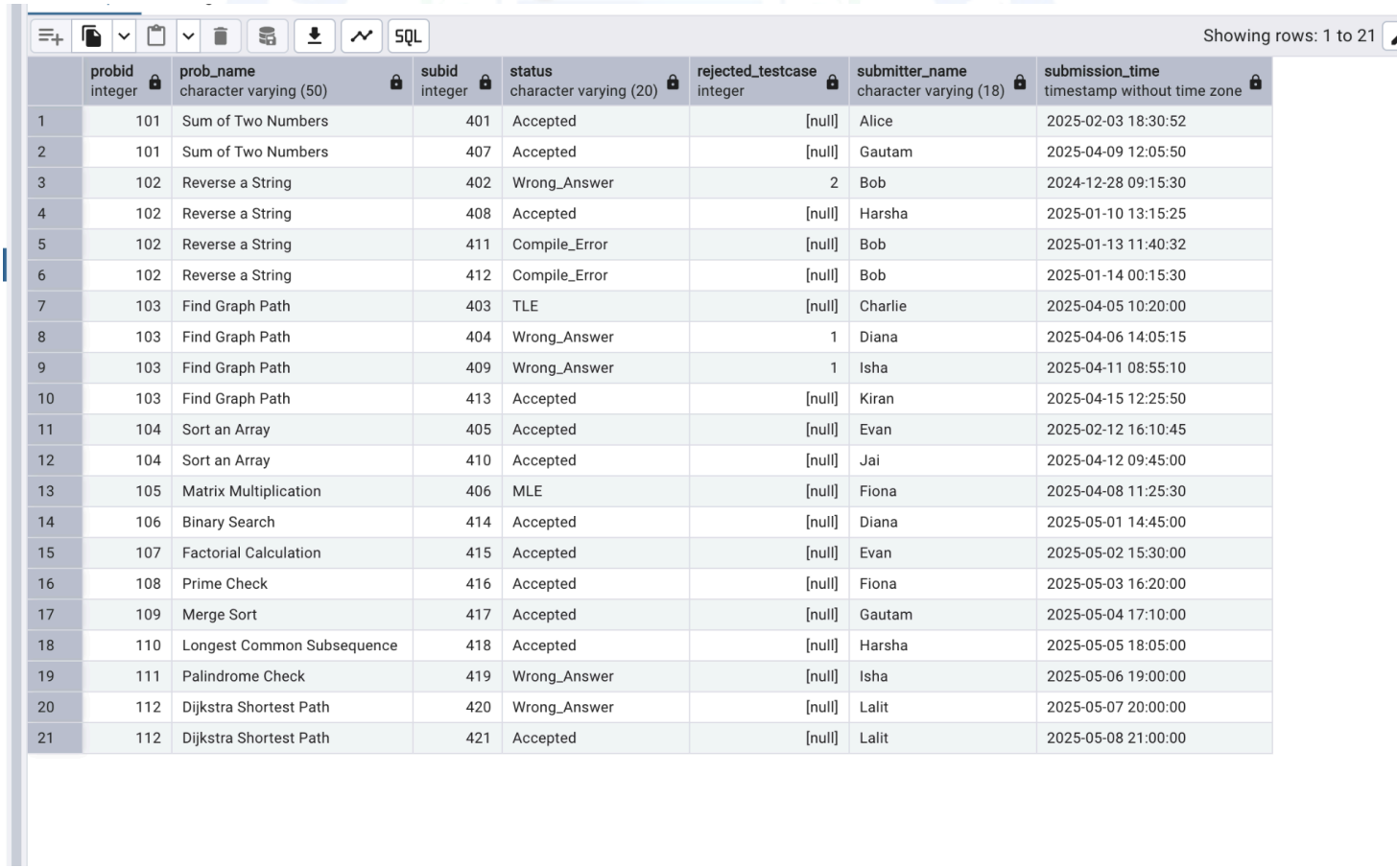
userid	name	email	phone_no	rating	reg_date	country	problems_solved	contests_participated	number_of_friends	favorite_problems	avg_runtime_s	avg_memory_mb	most_used_language
1	Alice	alice@example.com	1234567890	1500	2024-12-08 00:00:00	USA	1	2	2	2	1.00000000000000000000	128.0000000000000000	C++
2	Bob	bob@example.com	2345678901	1300	2024-11-15 00:00:00	Canada	NULL	2	2	1	2.0000000000000000	256.0000000000000000	Java
3	Charlie	charlie@example.com	3456789012	1200	2025-01-05 00:00:00	UK	NULL	2	1	2	3.0000000000000000	128.0000000000000000	C++
4	Diana	diana@example.com	4567890123	1600	2023-07-12 00:00:00	USA	1	2	1	2	1.5000000000000000	192.0000000000000000	C++
5	Evan	evan@example.com	5678901234	1100	2024-03-20 00:00:00	Australia	2	2	NULL	1	1.00000000000000000000	128.0000000000000000	C++
6	Fiona	fiona@example.com	6789012345	1400	2024-10-31 00:00:00	Germany	1	2	NULL	NULL	3.0000000000000000	576.0000000000000000	C++
7	Gautam	gautam@sample.in	9012345678	1550	2024-08-09 00:00:00	India	2	2	2	2	1.5000000000000000	128.0000000000000000	C++
8	Harsha	harsha@sample.in	9123456780	1250	2024-09-30 00:00:00	India	2	2	1	1	1.5000000000000000	128.0000000000000000	C++
8	Harsha	harsha@sample.in	9123456780	1250	2024-09-30 00:00:00	India	2	2	1	1	1.5000000000000000	128.0000000000000000	Python
9	Isha	isha@sample.in	9234567801	1350	2024-12-25 00:00:00	India	NULL	2	1	1	1.00000000000000000000	128.0000000000000000	C++
10	Jai	jai@sample.in	9345678012	1500	2025-02-14 00:00:00	India	1	2	1	2	1.00000000000000000000	128.0000000000000000	C++
11	Kiran	kiran@sample.com	9456780123	1450	2023-11-03 00:00:00	USA	1	2	NULL	1	1.00000000000000000000	128.0000000000000000	C++
12	Lalit	lalit@sample.in	9567801234	1600	2025-03-01 00:00:00	India	1	2	NULL	1	3.0000000000000000	512.0000000000000000	C++

2. List of all problems with their status of submissions:

```
SELECT
  p.probid,
  p.prob_name,
  s.subid,
  s.status,
  s.wrong_testcaseno AS rejected_testcase,
  u.name AS submitter_name,
```

```
    s.submission_time
FROM problems p
LEFT JOIN submissions s ON p.probid = s.probid
LEFT JOIN users u ON s.userid = u.userid
ORDER BY p.probid ASC, s.subid ASC;
```

Output :



Showing rows: 1 to 21

	probid integer	prob_name character varying (50)	subid integer	status character varying (20)	rejected_testcase integer	submitter_name character varying (18)	submission_time timestamp without time zone
1	101	Sum of Two Numbers	401	Accepted	[null]	Alice	2025-02-03 18:30:52
2	101	Sum of Two Numbers	407	Accepted	[null]	Gautam	2025-04-09 12:05:50
3	102	Reverse a String	402	Wrong_Answer	2	Bob	2024-12-28 09:15:30
4	102	Reverse a String	408	Accepted	[null]	Harsha	2025-01-10 13:15:25
5	102	Reverse a String	411	Compile_Error	[null]	Bob	2025-01-13 11:40:32
6	102	Reverse a String	412	Compile_Error	[null]	Bob	2025-01-14 00:15:30
7	103	Find Graph Path	403	TLE	[null]	Charlie	2025-04-05 10:20:00
8	103	Find Graph Path	404	Wrong_Answer	1	Diana	2025-04-06 14:05:15
9	103	Find Graph Path	409	Wrong_Answer	1	Isha	2025-04-11 08:55:10
10	103	Find Graph Path	413	Accepted	[null]	Kiran	2025-04-15 12:25:50
11	104	Sort an Array	405	Accepted	[null]	Evan	2025-02-12 16:10:45
12	104	Sort an Array	410	Accepted	[null]	Jai	2025-04-12 09:45:00
13	105	Matrix Multiplication	406	MLE	[null]	Fiona	2025-04-08 11:25:30
14	106	Binary Search	414	Accepted	[null]	Diana	2025-05-01 14:45:00
15	107	Factorial Calculation	415	Accepted	[null]	Evan	2025-05-02 15:30:00
16	108	Prime Check	416	Accepted	[null]	Fiona	2025-05-03 16:20:00
17	109	Merge Sort	417	Accepted	[null]	Gautam	2025-05-04 17:10:00
18	110	Longest Common Subsequence	418	Accepted	[null]	Harsha	2025-05-05 18:05:00
19	111	Palindrome Check	419	Wrong_Answer	[null]	Isha	2025-05-06 19:00:00
20	112	Dijkstra Shortest Path	420	Wrong_Answer	[null]	Lalit	2025-05-07 20:00:00
21	112	Dijkstra Shortest Path	421	Accepted	[null]	Lalit	2025-05-08 21:00:00

3. **Participant & Winner Statistics for contests with the users who designed them:**

SQL Query :

```
SELECT
  c.contest_name,
  c.contest_div       AS division,
  (
    SELECT string_agg(u2.name, ', ')
```

```
   FROM CodingPlatform.written_by wb2
   JOIN CodingPlatform.users u2
     ON wb2.userid = u2.userid
   WHERE wb2.contestid = c.contestid
  ) AS designers,
  (
   SELECT COUNT(*)
   FROM CodingPlatform.participates_in pi2
   WHERE pi2.contestid = c.contestid
  ) AS participant_count,

  u.name          AS winner_name,
  u.userid,
  pi.score,
  pi.time_spent_hrs

FROM CodingPlatform.contests c
JOIN CodingPlatform.participates_in pi
  ON c.contestid = pi.contestid
JOIN CodingPlatform.users u
  ON pi.userid = u.userid

WHERE pi.rank = 1

ORDER BY c.contest_date_time DESC;
```

OUTPUT :

Data Output | Messages | Notifications

	contest_name text	division integer	designers text	participant_count bigint	winner_name character varying (18)	userid integer	score integer	time_spent_hrs integer
1	New Year Sprint	1	Diana	3	Alice	1	99	2
2	Winter Bash	3	Alice, Jai	2	Harsha	8	92	2
3	Summer Showd...	4	Bob, Lalit	2	Jai	10	90	2
4	Tech Fiesta	2	Charlie, Isha	4	Diana	4	93	2
5	Autumn Cup	2	Charlie, Diana, Harsha	3	Evan	5	88	1
6	Spring Challenge	1	Evan, Fiona	10	Alice	1	100	2

4. **THE LEADERBOARD DISPLAY for any given contest (say contestid = 1):**

   SQL Query:

```
SELECT u.userid, u.name, u.rating, pi.rank, pi.score, pi.time_spent_hrs,
c.contest_name
FROM users u JOIN participates_in pi
ON u.userid = pi.userid JOIN contests c
ON pi.contestid = c.contestid
WHERE pi.contestid = 1
ORDER BY pi.rank ASC;
```

Output :

Data Output | Messages | Notifications

Showing r

	userid integer	name character varying (18)	rating integer	rank integer	score integer	time_spent_hrs integer	contest_name text
1	1	Alice	1500	1	100	2	Spring Challenge
2	2	Bob	1300	2	95	2	Spring Challenge
3	3	Charlie	1200	3	90	2	Spring Challenge
4	4	Diana	1600	4	85	2	Spring Challenge
5	7	Gautam	1550	5	80	2	Spring Challenge
6	8	Harsha	1250	6	75	2	Spring Challenge
7	9	Isha	1350	7	70	2	Spring Challenge
8	10	Jai	1500	8	65	2	Spring Challenge
9	11	Kiran	1450	9	60	2	Spring Challenge
10	12	Lalit	1600	10	55	2	Spring Challenge

5. **Problems-Discussions-Posts Count:**

   Query :

```
SELECT
   p.prob_name,
   d.topic,
   COUNT(p2.postid) AS post_count
FROM
   discussions d
JOIN
   about a ON d.discid = a.discid
JOIN
   problems p ON a.probid = p.probid
LEFT JOIN
```

```
    posts p2 ON d.discid = p2.discid
GROUP BY
    p.prob_name, d.topic
ORDER BY
    post_count DESC, p.prob_name;
```

OUTPUT :

Data Output | Messages | Notifications

	prob_name character varying (50)	topic character varying (50)	post_count bigint
1	Find Graph Path	Graph Path Problem - Clarifications	2
2	Reverse a String	Reversing Strings: Tips & Tricks	2
3	Sum of Two Numbers	Discussion on Sum of Two Numb...	2
4	Binary Search	Matrix Multiplication Q&A	1
5	Factorial Calculation	Matrix Multiplication Q&A	1
6	Matrix Multiplication	Matrix Multiplication Q&A	1
7	Sort an Array	Sorting Algorithms Comparison	1

END

➢We are extremely grateful to our professor - PM Jat Sir for giving us the opportunity to work on such an enriching project and also making us equipped with the requisite set of skills and knowledge throughout the semester. We would also like to extend this gratitude to our respected TAs for their guidance and support during the entire span of the project.

Thank You !!!