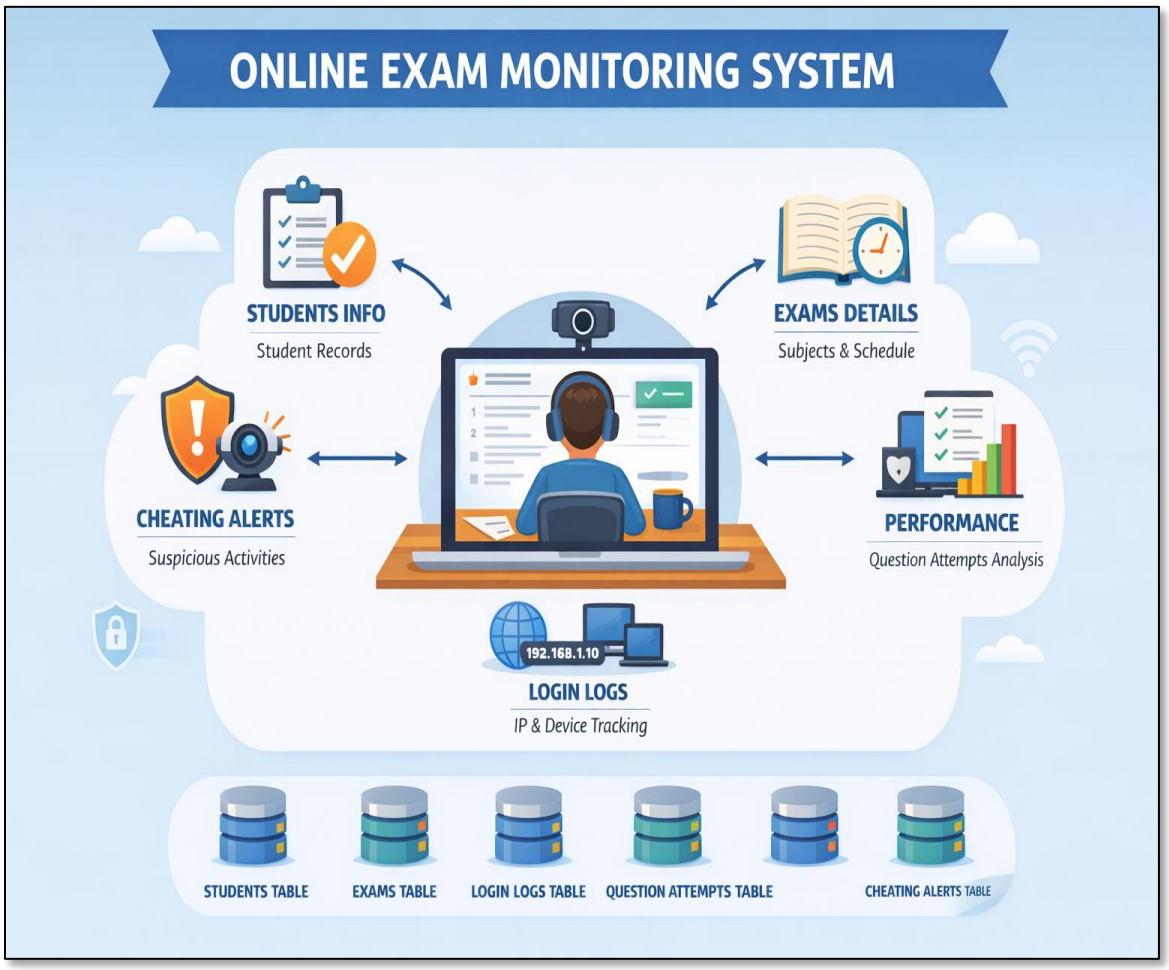
ONLINE EXAM MONITORING SYSTEM



NAME: Aniket Sunil Desale

INTRODUCTION

Online examinations have become increasingly common in modern education systems, making exam security and monitoring very important. The Online Exam Monitoring System is designed to conduct and supervise online exams in an efficient and secure manner. This system helps in tracking student details, exam schedules, login activities, and question attempts while ensuring fairness during the examination process.

This project uses **SQL** and relational database concepts to store and manage exam-related data such as students, exams, login logs, and cheating alerts. By organizing data using **primary** and **foreign** keys, the system ensures data accuracy and integrity. The Online Exam Monitoring System helps institutions detect suspicious activities, analyze student performance, and maintain transparency, making it a reliable solution for online examination management.

AIM OF THE PROJECT

- **To Develop an Online Exam Monitoring System:**

To design and implement a database-driven system that efficiently conducts and monitors online examinations.

- **To Ensure Secure and Fair Examinations:**

To monitor student activities during exams and detect any suspicious or cheating behavior.

• To Manage Exam and Student Data Effectively:

To store, organize, and maintain student details, exam schedules, and performance records accurately.

• To Analyze Student Performance:

To track question attempts and correctness to evaluate student performance during examinations.

• To Improve Transparency in Online Exams:

To provide a reliable system that ensures fairness, accountability, and integrity in the online examination process.

OBJECTIVES

• Set up the Online Exam Monitoring Database:

Design and create a structured database to store information related to students, exams, login logs, question attempts, and cheating alerts.

• Manage Exam and Student Records:

Maintain accurate records of student details, exam schedules, and exam duration for smooth exam management.

• Monitor Login and Exam Activities:

Track student login time, logout time, IP address, and device information during online examinations.

• Record Question Attempts and Performance:

Store details of each question attempt, including time taken and correctness of answers, to analyze student performance.

• Detect and Log Cheating Activities:

Identify and record suspicious activities such as tab switching, multiple faces, or mobile phone detection during exams.

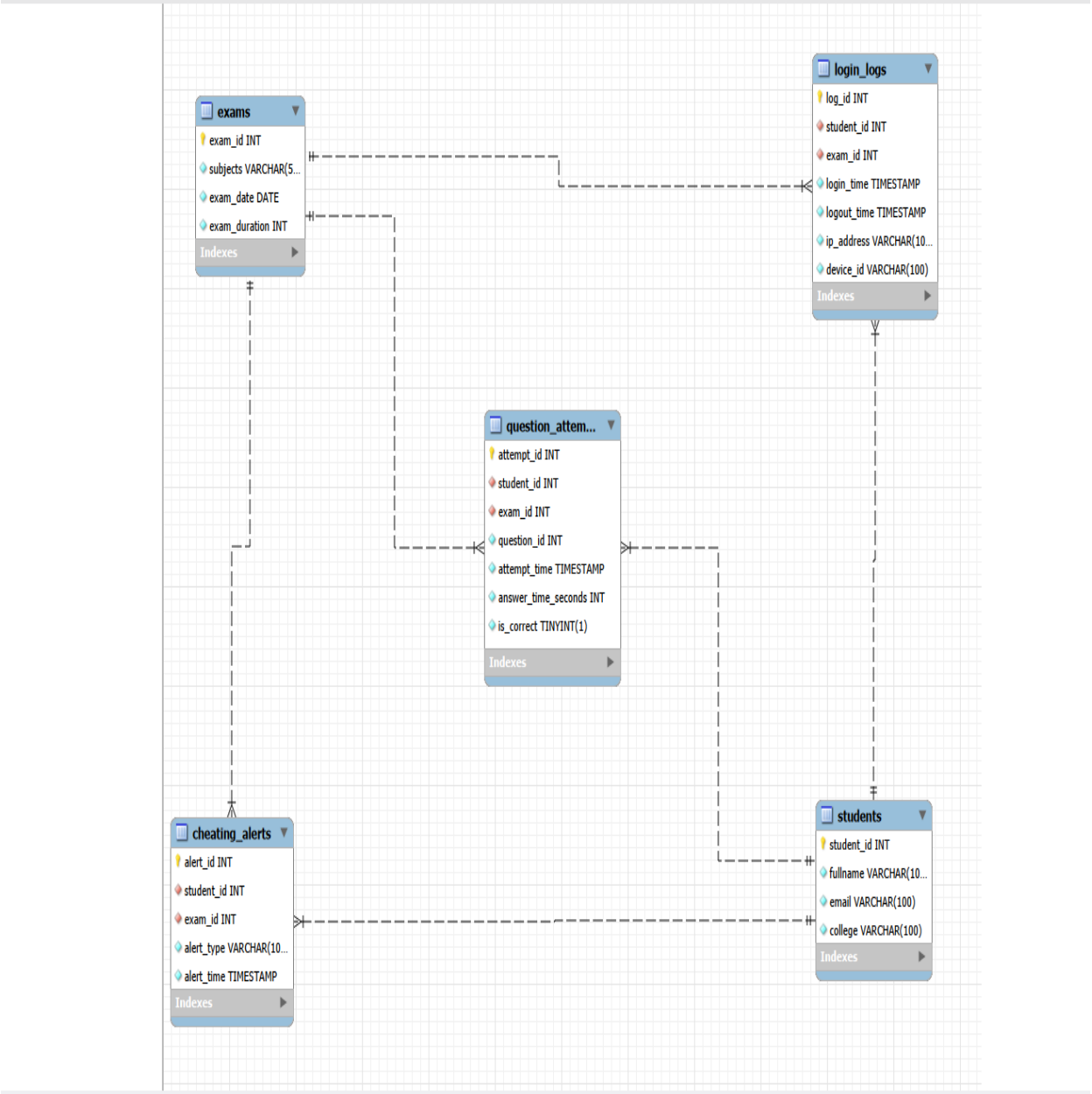
• Ensure Data Accuracy and Integrity:

Use primary keys, foreign keys, and constraints to maintain reliable and consistent data.

• Support SQL-Based Analysis:

Enable the use of SQL queries to retrieve meaningful insights for monitoring, evaluation, and reporting purposes.

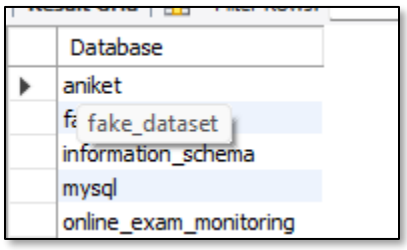
ER DIAGRAM



Creating Database:

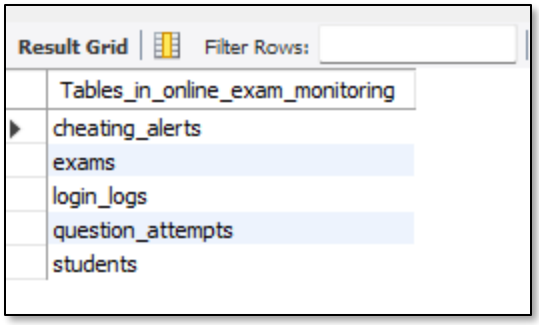
Create database online_exam_monitoring;

Use online_exam_monitoring;



Tables in Project:

show tables in online_exam_monitoring;



DATA DEFINITION LANGUAGE (DDL) ;

Creating Tables:

1) Students table

Create table students (

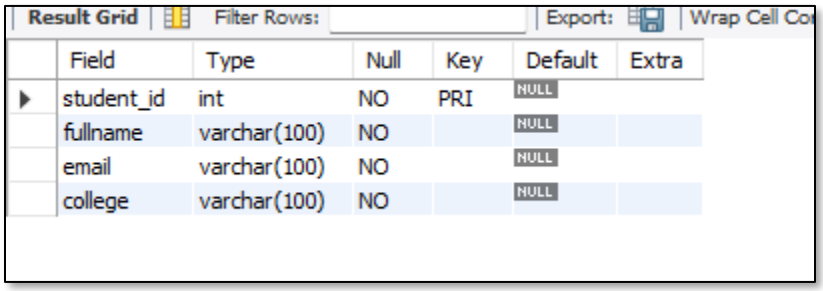
student_id int primary key not null,

fullname varchar(100) not null,

email varchar(100) not null,

college varchar(100) not null

);



Result Grid | Filter Rows: | Export: | Wrap Cell Co

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	
fullname	varchar(100)	NO		NULL	
email	varchar(100)	NO		NULL	
college	varchar(100)	NO		NULL	

2) Exams table

Create table exams (

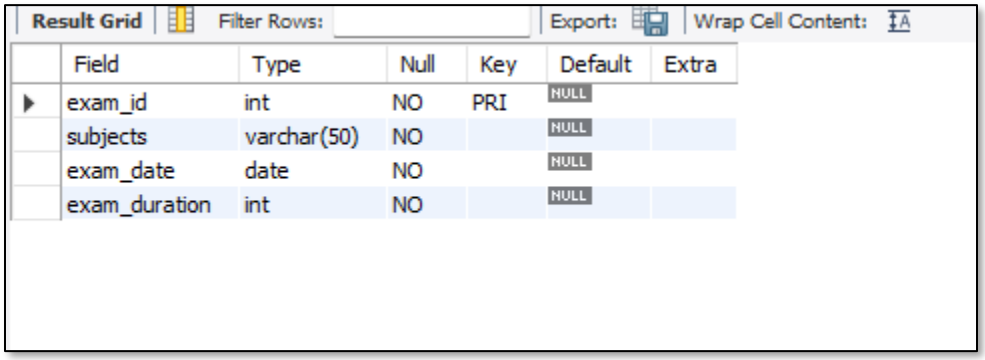
exam_id int primary key not null,

subjects varchar(50) not null,

exam_date date not null,

exam_duration int not null

);

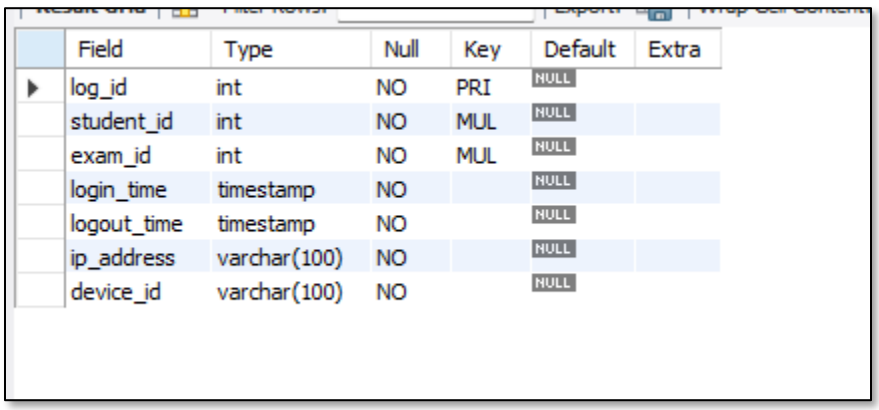


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
exam_id	int	NO	PRI	NULL	
subjects	varchar(50)	NO		NULL	
exam_date	date	NO		NULL	
exam_duration	int	NO		NULL	

3) Login logs table

```
Create table login_logs (
log_id int primary key not null,
student_id int not null,
exam_id int not null,
login_time timestamp not null,
logout_time timestamp not null,
ip_address varchar(100) not null,
device_id varchar(100) not null,
foreign key (student_id) references students(student_id),
foreign key (exam_id) references exams(exam_id)
);
```

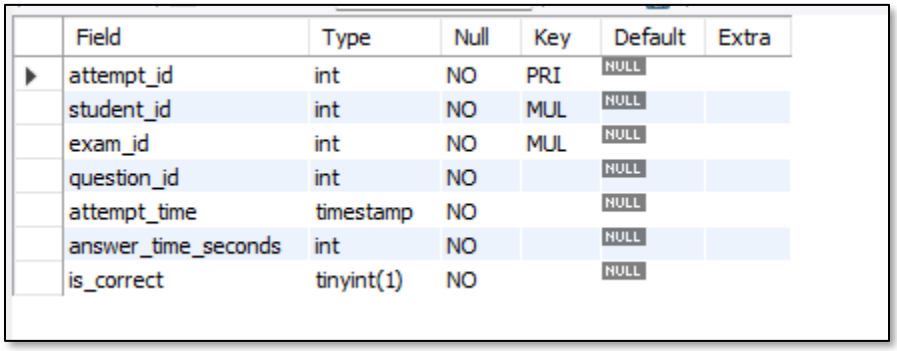


Field	Type	Null	Key	Default	Extra
log_id	int	NO	PRI	NULL	
student_id	int	NO	MUL	NULL	
exam_id	int	NO	MUL	NULL	
login_time	timestamp	NO		NULL	
logout_time	timestamp	NO		NULL	
ip_address	varchar(100)	NO		NULL	
device_id	varchar(100)	NO		NULL	

4) Question attempts table

```
Create table question_attempts (
```

```
attempt_id int primary key not null,
student_id int not null,
exam_id int not null,
question_id int not null,
attempt_time timestamp not null,
answer_time_seconds int not null,
is_correct boolean not null,
foreign key (student_id) references students(student_id),
foreign key (exam_id) references exams(exam_id)
);
```



Field	Type	Null	Key	Default	Extra
attempt_id	int	NO	PRI	NULL	
student_id	int	NO	MUL	NULL	
exam_id	int	NO	MUL	NULL	
question_id	int	NO		NULL	
attempt_time	timestamp	NO		NULL	
answer_time_seconds	int	NO		NULL	
is_correct	tinyint(1)	NO		NULL	

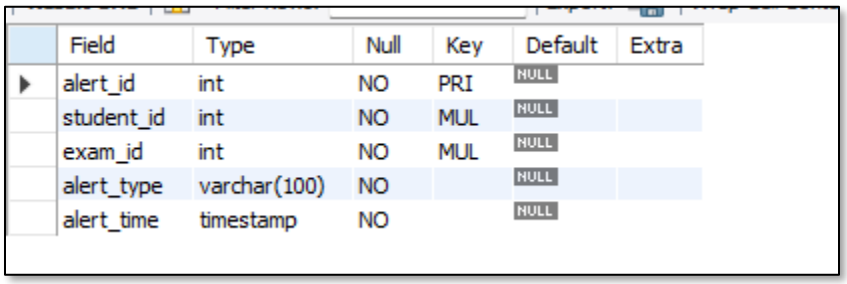
5) Cheating alerts table

```
Create table cheating_alerts (
alert_id int primary key not null,
student_id int not null,
exam_id int not null,
alert_type varchar(100) not null,
alert_time timestamp not null,
```

foreign key (student_id) references students(student_id),

foreign key (exam_id) references exams(exam_id)

);



Field	Type	Null	Key	Default	Extra
alert_id	int	NO	PRI	NULL	
student_id	int	NO	MUL	NULL	
exam_id	int	NO	MUL	NULL	
alert_type	varchar(100)	NO		NULL	
alert_time	timestamp	NO		NULL	

DATA MANUPULATION LANGUAGE (DML) :

Inserting values in Tables:

1) Students table

Insert into

students (student_id, fullname, email, college)

values

(1, 'Amit Sharma', 'amit.sharma@gmail.com', 'Csm University'),

(2, 'Neha Verma', 'neha.verma@gmail.com', 'Csm University'),

(3, 'Rahul Patil', 'rahul.patil@gmail.com', 'Csm University'),

(4, 'Sneha Kulkarni', 'sneha.k@gmail.com', 'Csm University'),

(5, 'Rohit Singh', 'rohit.singh@gmail.com', 'Csm University'),

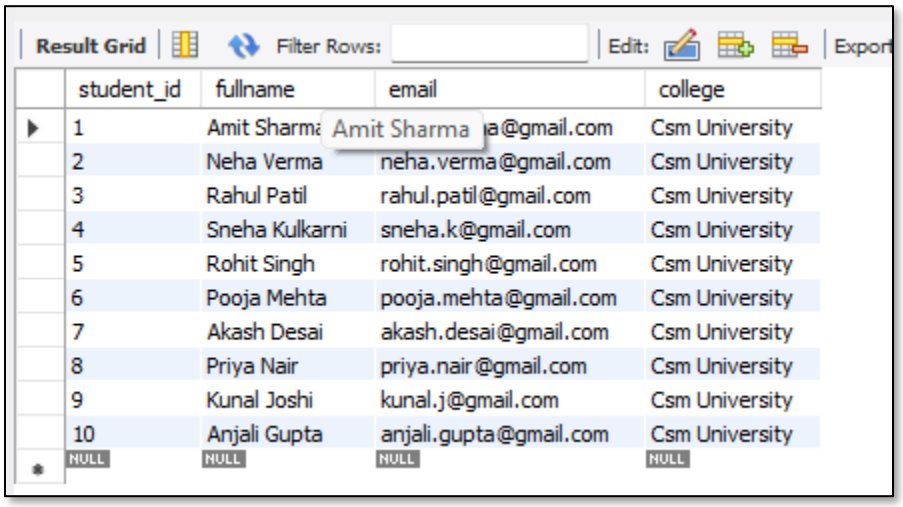
(6, 'Pooja Mehta', 'pooja.mehta@gmail.com', 'Csm University'),

(7, 'Akash Desai', 'akash.desai@gmail.com', 'Csm University'),

(8, 'Priya Nair', 'priya.nair@gmail.com', 'Csm University'),

(9, 'Kunal Joshi', 'kunal.j@gmail.com', 'Csm University'),

(10, 'Anjali Gupta', 'anjali.gupta@gmail.com', 'Csm University');



Result Grid | Filter Rows: | Edit: | Export

student_id	fullname	email	college
1	Amit Sharma	Amit Sharma a@gmail.com	Csm University
2	Neha Verma	neha.verma@gmail.com	Csm University
3	Rahul Patil	rahul.patil@gmail.com	Csm University
4	Sneha Kulkarni	sneha.k@gmail.com	Csm University
5	Rohit Singh	rohit.singh@gmail.com	Csm University
6	Pooja Mehta	pooja.mehta@gmail.com	Csm University
7	Akash Desai	akash.desai@gmail.com	Csm University
8	Priya Nair	priya.nair@gmail.com	Csm University
9	Kunal Joshi	kunal.j@gmail.com	Csm University
10	Anjali Gupta	anjali.gupta@gmail.com	Csm University
NULL	NULL	NULL	NULL

2) Exams table

Insert into

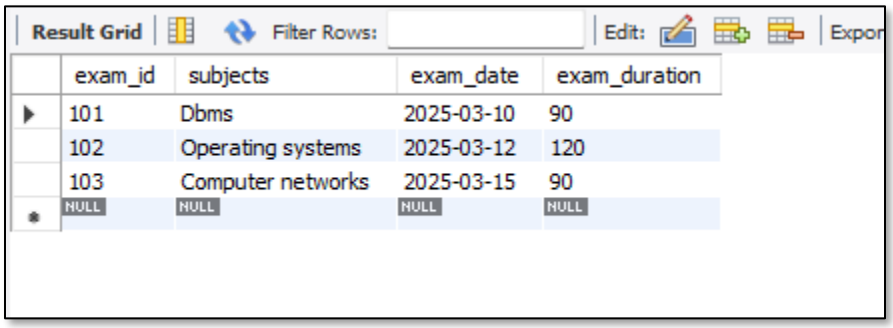
exams (exam_id, subjects, exam_date, exam_duration)

values

(101, 'Dbms', '2025-03-10', 90),

(102, 'Operating systems', '2025-03-12', 120),

(103, 'Computer networks', '2025-03-15', 90);

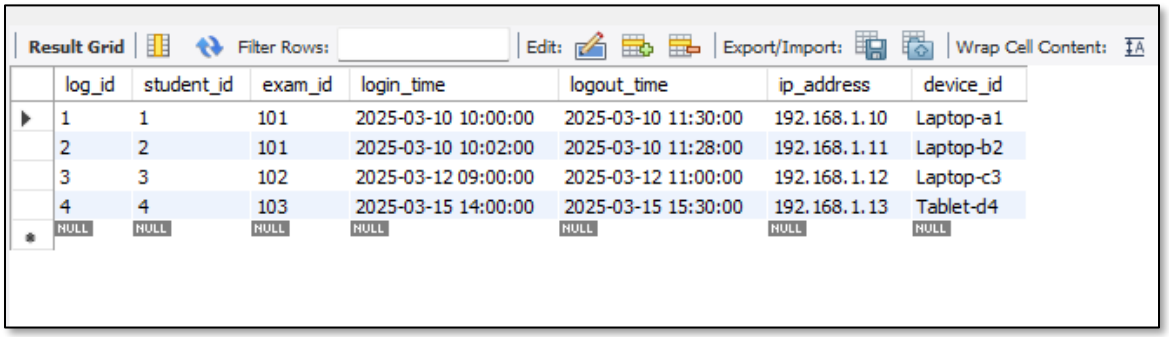


Result Grid | Filter Rows: | Edit: | Export

exam_id	subjects	exam_date	exam_duration
101	Dbms	2025-03-10	90
102	Operating systems	2025-03-12	120
103	Computer networks	2025-03-15	90
NULL	NULL	NULL	NULL

3) Login logs table

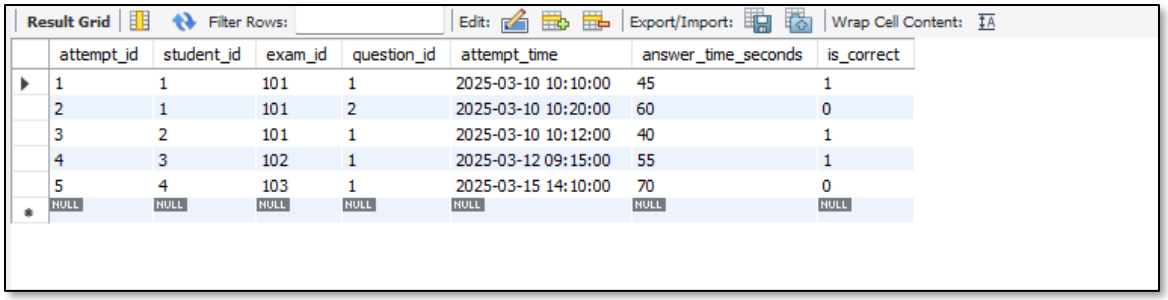
Insert into
login_logs
(log_id, student_id, exam_id, login_time, logout_time, ip_address, device_id)
values
(1, 1, 101, '2025-03-10 10:00:00', '2025-03-10 11:30:00', '192.168.1.10', 'Laptop-a1'),
(2, 2, 101, '2025-03-10 10:02:00', '2025-03-10 11:28:00', '192.168.1.11', 'Laptop-b2'),
(3, 3, 102, '2025-03-12 09:00:00', '2025-03-12 11:00:00', '192.168.1.12', 'Laptop-c3'),
(4, 4, 103, '2025-03-15 14:00:00', '2025-03-15 15:30:00', '192.168.1.13', 'Tablet-d4');



log_id	student_id	exam_id	login_time	logout_time	ip_address	device_id
1	1	101	2025-03-10 10:00:00	2025-03-10 11:30:00	192.168.1.10	Laptop-a1
2	2	101	2025-03-10 10:02:00	2025-03-10 11:28:00	192.168.1.11	Laptop-b2
3	3	102	2025-03-12 09:00:00	2025-03-12 11:00:00	192.168.1.12	Laptop-c3
4	4	103	2025-03-15 14:00:00	2025-03-15 15:30:00	192.168.1.13	Tablet-d4
NULL	NULL	NULL	NULL	NULL	NULL	NULL

4) Question attempts table

Insert into
question_attempts (attempt_id, student_id, exam_id, question_id,
attempt_time, answer_time_seconds, is_correct)
values
(1, 1, 101, 1, '2025-03-10 10:10:00', 45, true),
(2, 1, 101, 2, '2025-03-10 10:20:00', 60, false),
(3, 2, 101, 1, '2025-03-10 10:12:00', 40, true),
(4, 3, 102, 1, '2025-03-12 09:15:00', 55, true),
(5, 4, 103, 1, '2025-03-15 14:10:00', 70, false);

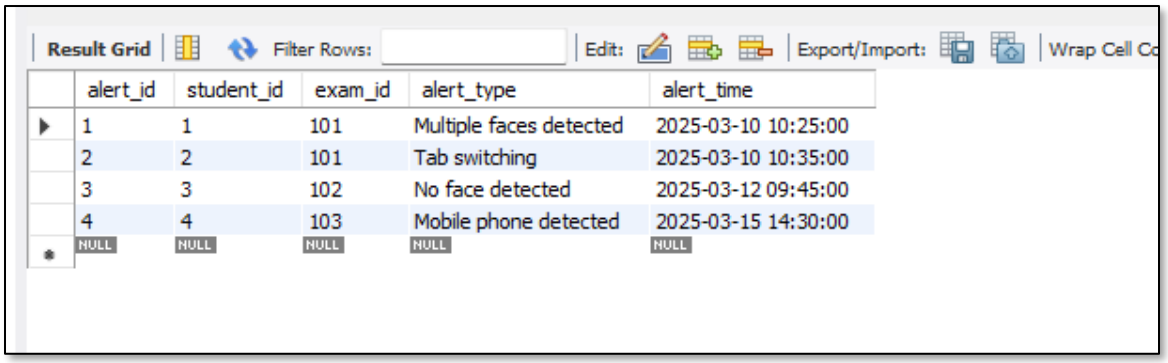


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

attempt_id	student_id	exam_id	question_id	attempt_time	answer_time_seconds	is_correct
1	1	101	1	2025-03-10 10:10:00	45	1
2	1	101	2	2025-03-10 10:20:00	60	0
3	2	101	1	2025-03-10 10:12:00	40	1
4	3	102	1	2025-03-12 09:15:00	55	1
5	4	103	1	2025-03-15 14:10:00	70	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

5) Cheating alerts table

```
Insert into
 cheating_alerts
(alert_id, student_id, exam_id, alert_type, alert_time)
values
(1, 1, 101, 'Multiple faces detected', '2025-03-10 10:25:00'),
(2, 2, 101, 'Tab switching', '2025-03-10 10:35:00'),
(3, 3, 102, 'No face detected', '2025-03-12 09:45:00'),
(4, 4, 103, 'Mobile phone detected', '2025-03-15 14:30:00');
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

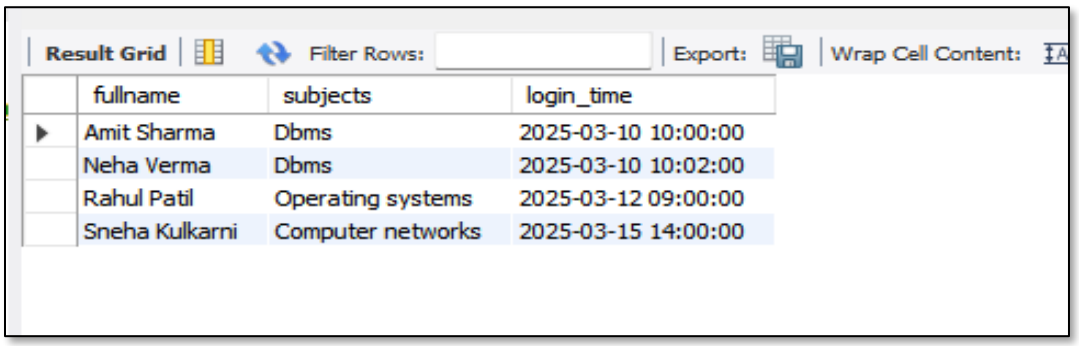
alert_id	student_id	exam_id	alert_type	alert_time
1	1	101	Multiple faces detected	2025-03-10 10:25:00
2	2	101	Tab switching	2025-03-10 10:35:00
3	3	102	No face detected	2025-03-12 09:45:00
4	4	103	Mobile phone detected	2025-03-15 14:30:00
NULL	NULL	NULL	NULL	NULL

SQL QUESTIONS & ANSWERS

Join Based Questions

1. Display student name, subject, and login time for each exam attempt.

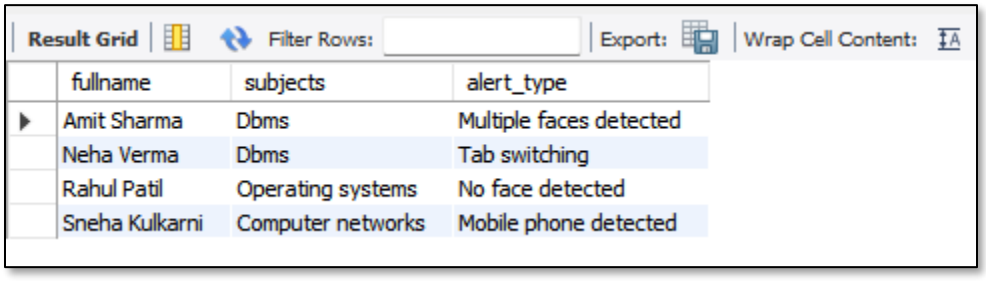
Answer:
Select s.fullname, e.subjects, l.login_time
from students as s
join login_logs as l on s.student_id = l.student_id
join exams as e on l.exam_id = e.exam_id;



fullname	subjects	login_time
Amit Sharma	Dbms	2025-03-10 10:00:00
Neha Verma	Dbms	2025-03-10 10:02:00
Rahul Patil	Operating systems	2025-03-12 09:00:00
Sneha Kulkarni	Computer networks	2025-03-15 14:00:00

2. List students who received cheating alerts along with alert type and exam subject.

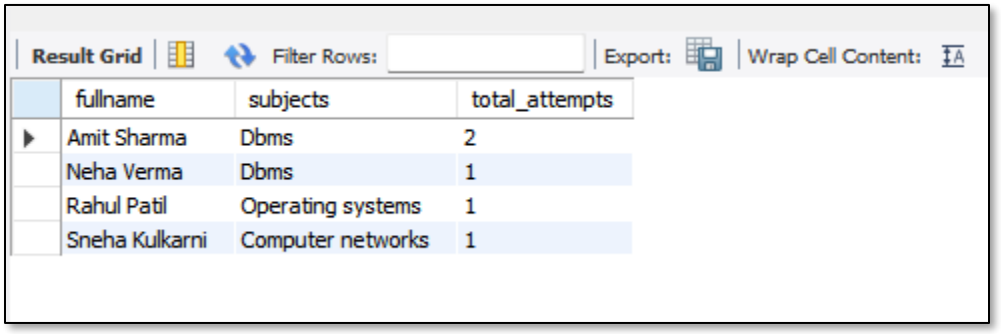
Answer:
Select s.fullname, e.subjects, c.alert_type
from students as s
join cheating_alerts as c on s.student_id = c.student_id
join exams as e on c.exam_id = e.exam_id;



fullname	subjects	alert_type
Amit Sharma	Dbms	Multiple faces detected
Neha Verma	Dbms	Tab switching
Rahul Patil	Operating systems	No face detected
Sneha Kulkarni	Computer networks	Mobile phone detected

3. Show student name, subject, and total number of questions attempted.

Answer:
Select s.fullname, e.subjects, count(q.attempt_id) as total_attempts
from students as s
join question_attempts as q on s.student_id = q.student_id
join exams as e on q.exam_id = e.exam_id
Group by s.fullname, e.subjects;

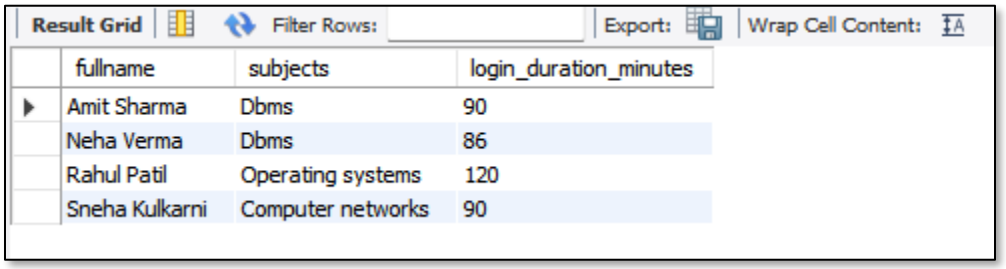


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fullname	subjects	total_attempts
Amit Sharma	Dbms	2
Neha Verma	Dbms	1
Rahul Patil	Operating systems	1
Sneha Kulkarni	Computer networks	1

4. Display login duration of students for each exam.

Answer:
Select s.fullname, e.subjects,
Timestampdiff(minute, l.login_time, l.logout_time) as login_duration_minutes
from students as s
join login_logs l on s.student_id = l.student_id
join exams as e on l.exam_id = e.exam_id;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fullname	subjects	login_duration_minutes
Amit Sharma	Dbms	90
Neha Verma	Dbms	86
Rahul Patil	Operating systems	120
Sneha Kulkarni	Computer networks	90

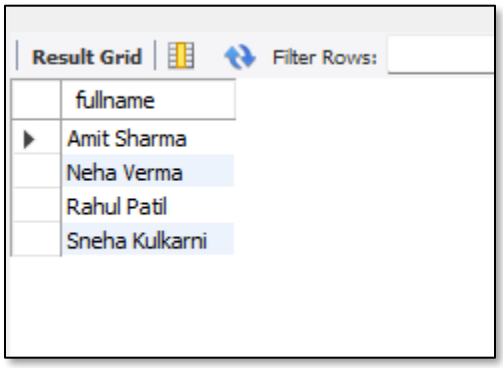
5. Find students who attempted questions and also received cheating alerts.

Answer:
Select distinct s.fullname
from students as s
join question_attempts as q on s.student_id = q.student_id
join cheating_alerts as c on s.student_id = c.student_id;

Subquery Based Questions

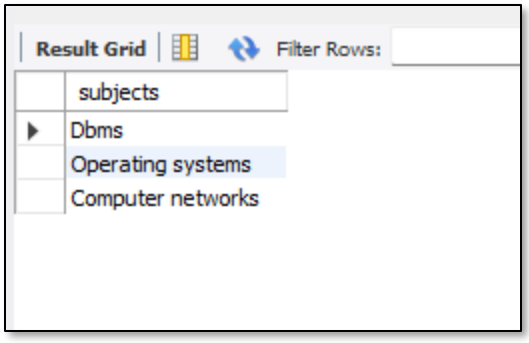
6. Find students who have received at least one cheating alert.

Answer:
Select fullname
from students
where student_id in (
Select student_id from cheating_alerts
);



7. Display exams where cheating alerts were recorded.

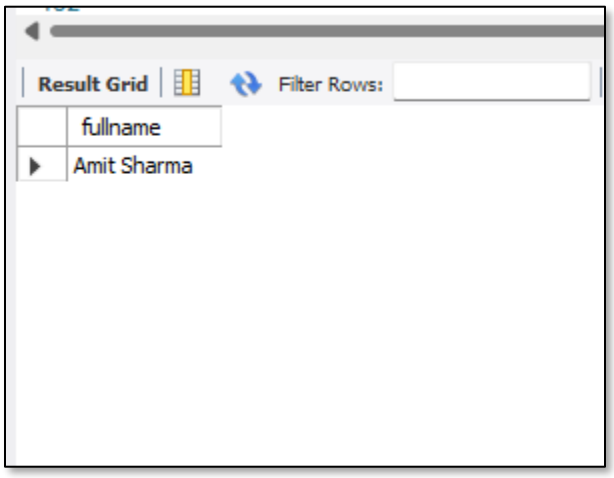
Answer:
Select subjects
from exams
where exam_id in (
Select exam_id from cheating_alerts
);



8. Find students who attempted more questions than the average.

Answer:

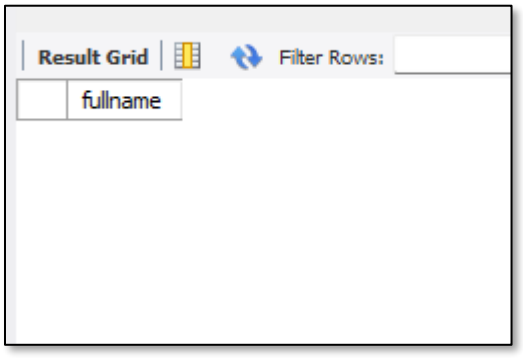
```
Select fullname
from students
where student_id in (
 Select student_id
 from question_attempts
 Group by student_id
 Having count(*) >
  (Select avg(cnt)
   from (
    Select count(*) as cnt
    from question_attempts
    Group by student_id
   ) as avg_table)
);
```



9. List students who logged in but never attempted any question.

Answer:

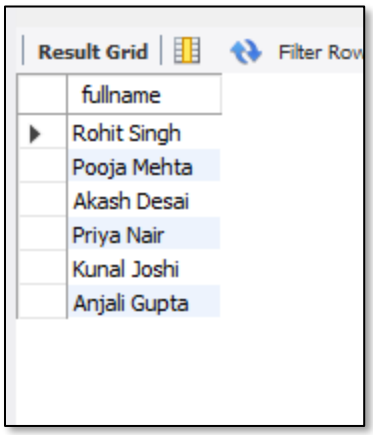
```
Select fullname
from students
where student_id in (
  Select student_id from login_logs  )
and student_id not in (
  Select student_id from question_attempts
);
```



10. Find students who never received any cheating alert.

Answer:

```
Select fullname
from students
where student_id not in (
  Select student_id from cheating_alerts
);
```

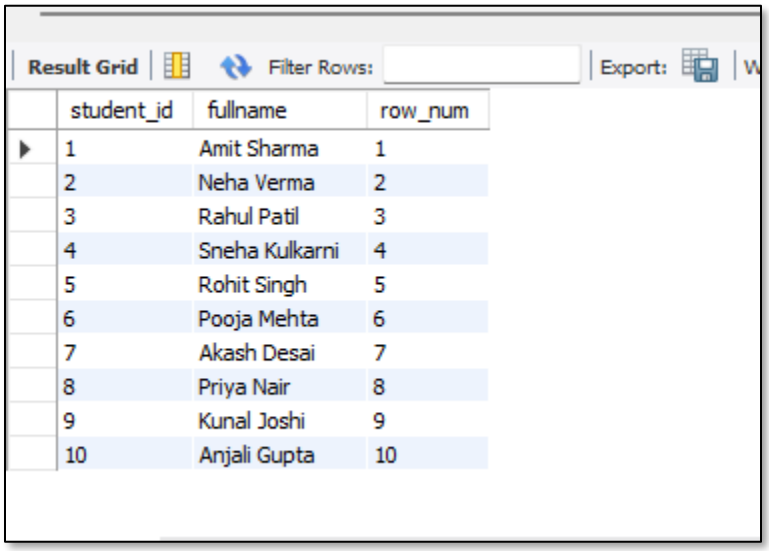


Window Functions:

Q1) Write a SQL query to assign a row number to each student based on student_id.

Answer:

```
Select student_id, fullname,

Row_number() over (order by student_id) as row_num

from students;
```



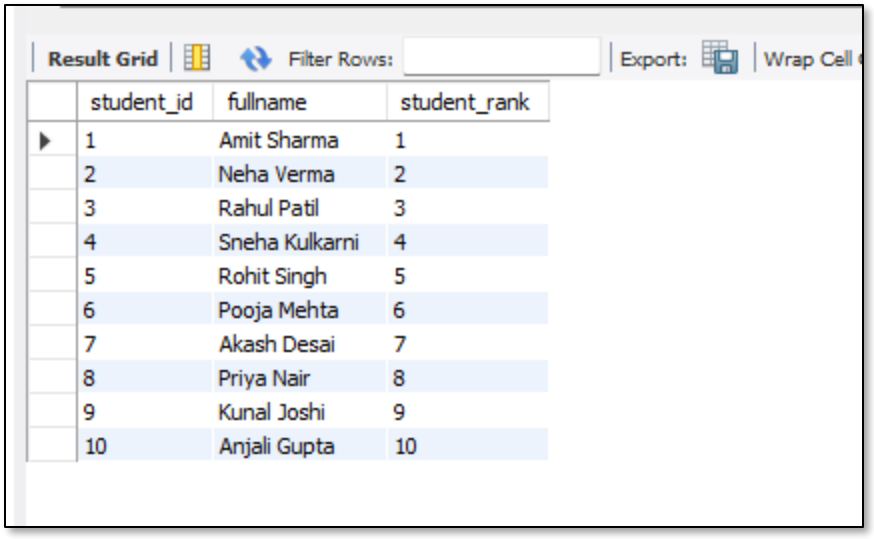
Result Grid | Filter Rows: | Export:

student_id	fullname	row_num
1	Amit Sharma	1
2	Neha Verma	2
3	Rahul Patil	3
4	Sneha Kulkarni	4
5	Rohit Singh	5
6	Pooja Mehta	6
7	Akash Desai	7
8	Priya Nair	8
9	Kunal Joshi	9
10	Anjali Gupta	10

Q2) Write a SQL query to display students with ranking based on student_id using RANK().

Answer:

```
Select student_id, fullname,

Rank() over (order by student_id) as student_rank

from students;
```



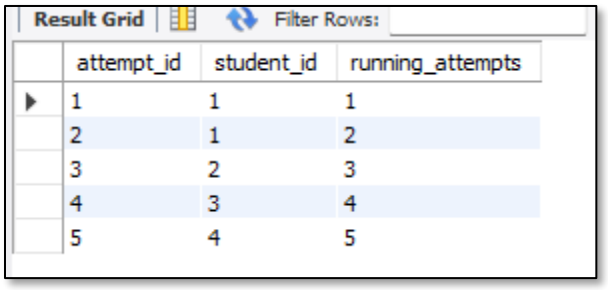
Result Grid | Filter Rows: | Export: | Wrap Cell

student_id	fullname	student_rank
1	Amit Sharma	1
2	Neha Verma	2
3	Rahul Patil	3
4	Sneha Kulkarni	4
5	Rohit Singh	5
6	Pooja Mehta	6
7	Akash Desai	7
8	Priya Nair	8
9	Kunal Joshi	9
10	Anjali Gupta	10

Q3) Write a SQL query to show running total of question attempts based on attempt_id.

Answer:

```
Select attempt_id, student_id,
Count(*) over (order by attempt_id) as running_attempts
from question_attempts;
```



Result Grid | Filter Rows:

attempt_id	student_id	running_attempts
1	1	1
2	1	2
3	2	3
4	3	4
5	4	5

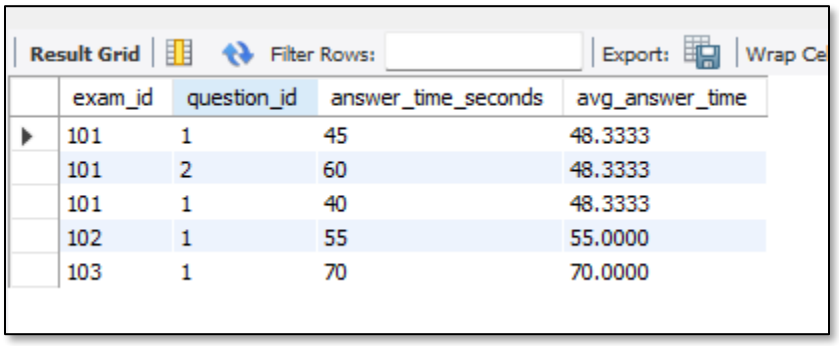
Q4) Write a SQL query to find the average answer time for each exam using a window function.

Answer:

```
Select exam_id, question_id, answer_time_seconds,
```

Avg(answer_time_seconds) over (partition by exam_id) as avg_answer_time

from question_attempts;



Result Grid | Filter Rows: | Export: | Wrap Ce

exam_id	question_id	answer_time_seconds	avg_answer_time
101	1	45	48.3333
101	2	60	48.3333
101	1	40	48.3333
102	1	55	55.0000
103	1	70	70.0000

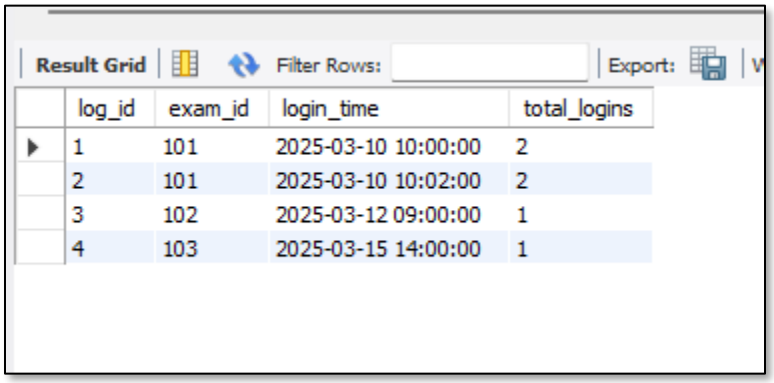
Q5) Write a SQL query to display login logs along with total login count per exam.

Answer:

Select log_id, exam_id, login_time,

Count(*) over (partition by exam_id) as total_logins

from login_logs**;**



Result Grid | Filter Rows: | Export:

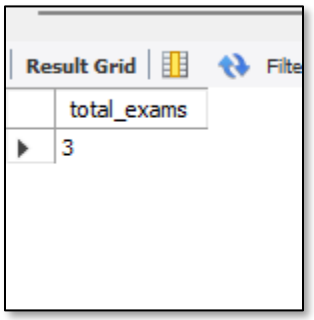
log_id	exam_id	login_time	total_logins
1	101	2025-03-10 10:00:00	2
2	101	2025-03-10 10:02:00	2
3	102	2025-03-12 09:00:00	1
4	103	2025-03-15 14:00:00	1

Extra Questions:-

Q1) Write a SQL query to find the total number of exams conducted.

Answer:

Select count(exam_id) as total_ exams from exams;

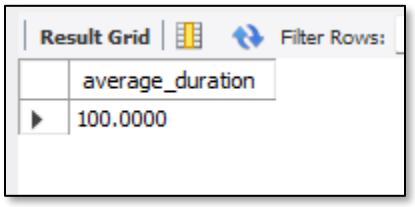


Q2) Write a SQL query to find the average duration of exams.

Answer:

Select avg(exam_duration) as average_duration

from exams;

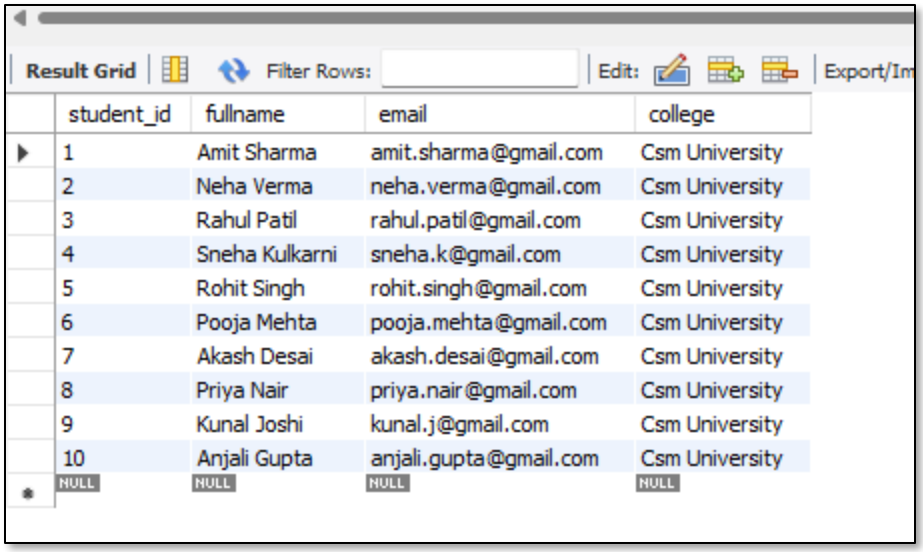


Q3) Write a SQL query to display students whose email ends with @gmail.com.

Answer:

Select * from students

Where email like '%@gmail.com';



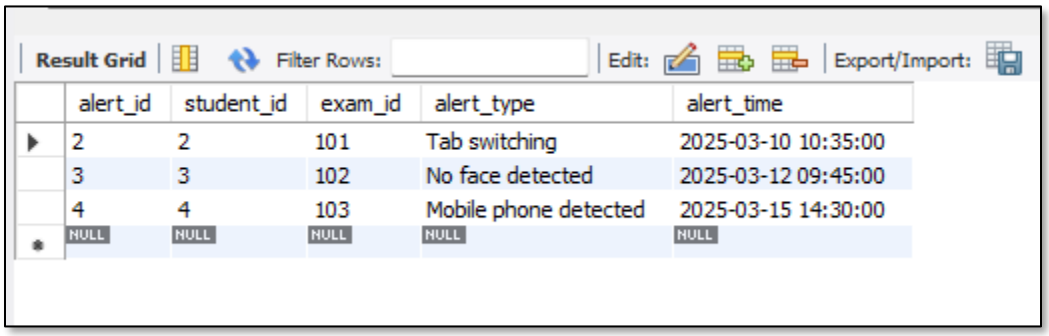
Result Grid | Filter Rows: | Edit: | Export/Im

student_id	fullname	email	college
1	Amit Sharma	amit.sharma@gmail.com	Csm University
2	Neha Verma	neha.verma@gmail.com	Csm University
3	Rahul Patil	rahul.patil@gmail.com	Csm University
4	Sneha Kulkarni	sneha.k@gmail.com	Csm University
5	Rohit Singh	rohit.singh@gmail.com	Csm University
6	Pooja Mehta	pooja.mehta@gmail.com	Csm University
7	Akash Desai	akash.desai@gmail.com	Csm University
8	Priya Nair	priya.nair@gmail.com	Csm University
9	Kunal Joshi	kunal.j@gmail.com	Csm University
10	Anjali Gupta	anjali.gupta@gmail.com	Csm University
NULL	NULL	NULL	NULL

Q1) Write a SQL query to display cheating alerts generated after 10:30 AM on 10 March 2025.

Answer:

```
Select * from   cheating_alerts

where   alert_time  >  '2025-03-10 10:30:00';
```



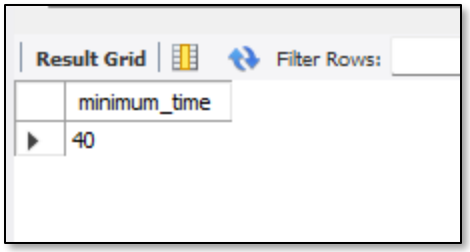
Result Grid | Filter Rows: | Edit: | Export/Import:

alert_id	student_id	exam_id	alert_type	alert_time
2	2	101	Tab switching	2025-03-10 10:35:00
3	3	102	No face detected	2025-03-12 09:45:00
4	4	103	Mobile phone detected	2025-03-15 14:30:00
NULL	NULL	NULL	NULL	NULL

Q5) Write a SQL query to find the minimum time taken to answer a question.

Answer:

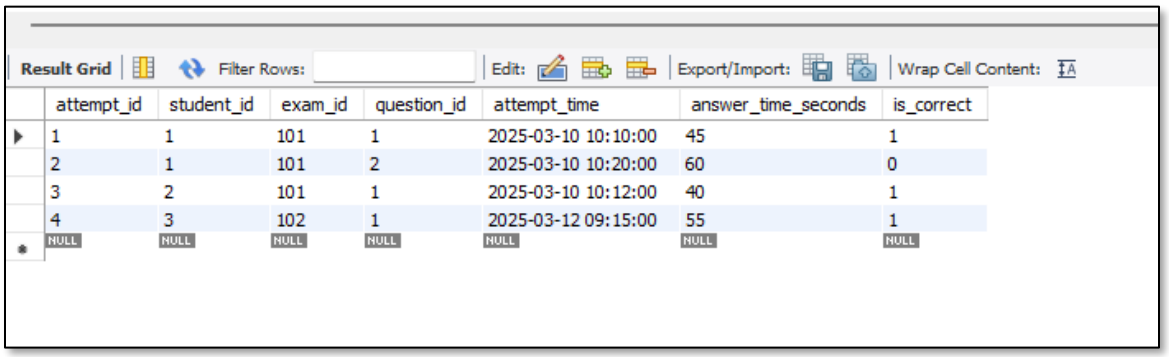
```
Select   min(answer_time_seconds)    as minimum_time

from    question_attempts;
```



Q6) Write a SQL query to display question attempts where answer time is between 40 and 60 seconds.

Answer:

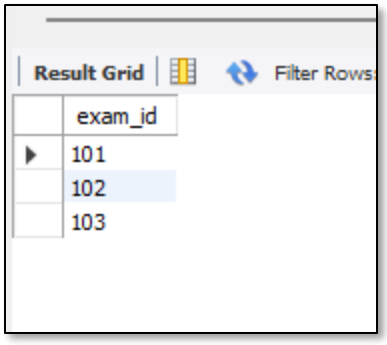
```
Select * from question_attempts

where answer_time_seconds between 40 and 60;
```



Q7) Write a SQL query to display distinct exam IDs from login logs.

Answer:

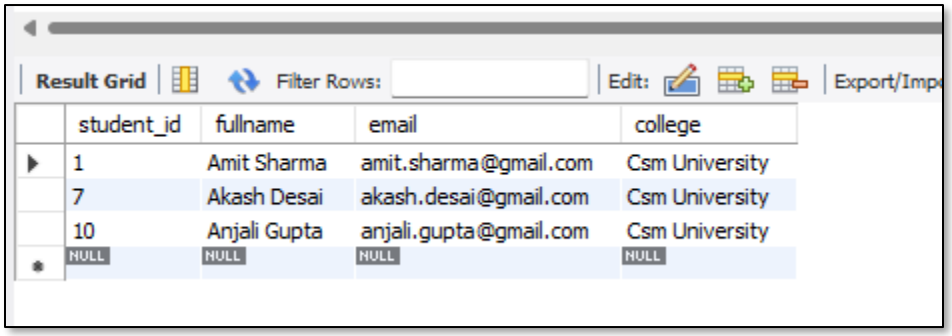
```
Select distinct exam_id

from login_logs;
```



Q8) Write a SQL query to display students whose name starts with the letter 'A'.

Answer:

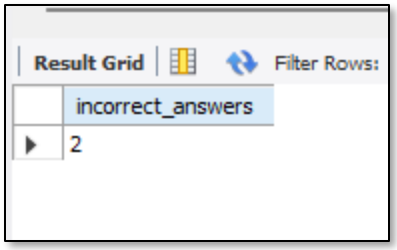
```
Select * from students
where fullname like 'A%';
```



Q9) Write a SQL query to find the total number of incorrect answers.

Answer:

```
Select count(*) as incorrect_answers
from question_attempts
where is_correct = false;
```

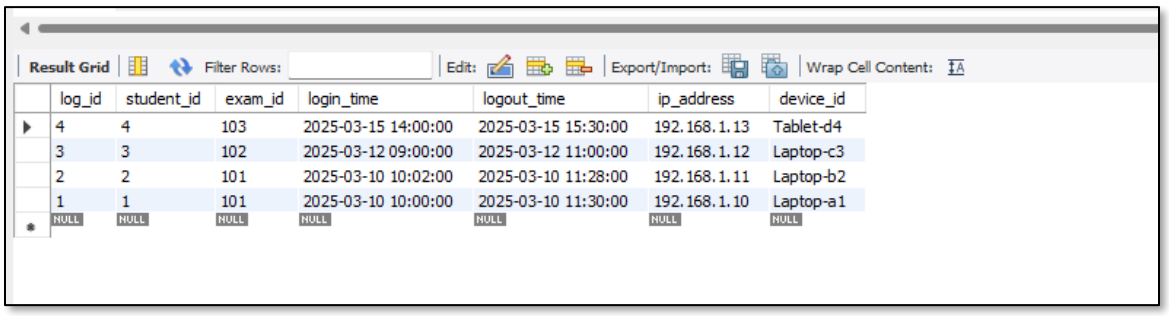


Q10) Write a SQL query to display login logs sorted by login time in descending order.

Answer:

Select * from login_logs

Order by login_time desc;



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

log_id	student_id	exam_id	login_time	logout_time	ip_address	device_id
4	4	103	2025-03-15 14:00:00	2025-03-15 15:30:00	192.168.1.13	Tablet-d4
3	3	102	2025-03-12 09:00:00	2025-03-12 11:00:00	192.168.1.12	Laptop-c3
2	2	101	2025-03-10 10:02:00	2025-03-10 11:28:00	192.168.1.11	Laptop-b2
1	1	101	2025-03-10 10:00:00	2025-03-10 11:30:00	192.168.1.10	Laptop-a1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

CONCLUSION

The **Online Exam Monitoring System** database is designed to manage and analyze online examinations in a structured and secure manner. This database effectively stores and organizes information related to students, exams, login activities, question attempts, and cheating alerts, which are critical components of any online examination platform.

In this system, relational database concepts are used to maintain data accuracy and integrity. Each table is created with **primary keys,** ensuring that every record is uniquely identified. Foreign keys are used to establish relationships between tables, which helps in maintaining referential integrity and prevents invalid data entries. This structure ensures that student activities are always linked to valid students and exams.

Various SQL operations are used to retrieve meaningful information from the database. The **SELECT** statement is used to fetch required data, while the **WHERE** clause helps in filtering records based on specific conditions such as dates, exam duration, correctness of answers, and alert time. Aggregate functions like **COUNT, AVG, MIN,** and **MAX** are used to perform calculations and generate summarized insights, such as total exams conducted, average exam duration, and incorrect answers.

The use of string functions and pattern matching with the **LIKE** operator allows efficient searching of student records, such as identifying email domains or names starting with specific letters. The **BETWEEN** operator simplifies range-based filtering, which is useful for analyzing answer times. The **DISTINCT** keyword helps in eliminating duplicate values, ensuring cleaner and more accurate outputs. Sorting of records using **ORDER BY** improves readability and analysis of time-based data like login logs.

Boolean values (**TRUE** and **FALSE)** are effectively used in the question_attempts table to represent whether an answer is correct or incorrect. This makes performance analysis simple and efficient. Timestamp and date fields play an important role in tracking login durations, exam schedules, and cheating alert timings, which are essential for monitoring and audit purposes.

Overall, this database system demonstrates how SQL is crucial for managing real-time online exam data. It ensures data consistency, security, and transparency, making it highly suitable for academic institutions. Understanding and implementing such SQL queries is extremely important for students, as it strengthens their knowledge of DBMS concepts, prepares them for practical exams and interviews, and helps them build real-world database-driven applications.