

O

RAWDA

REYAM

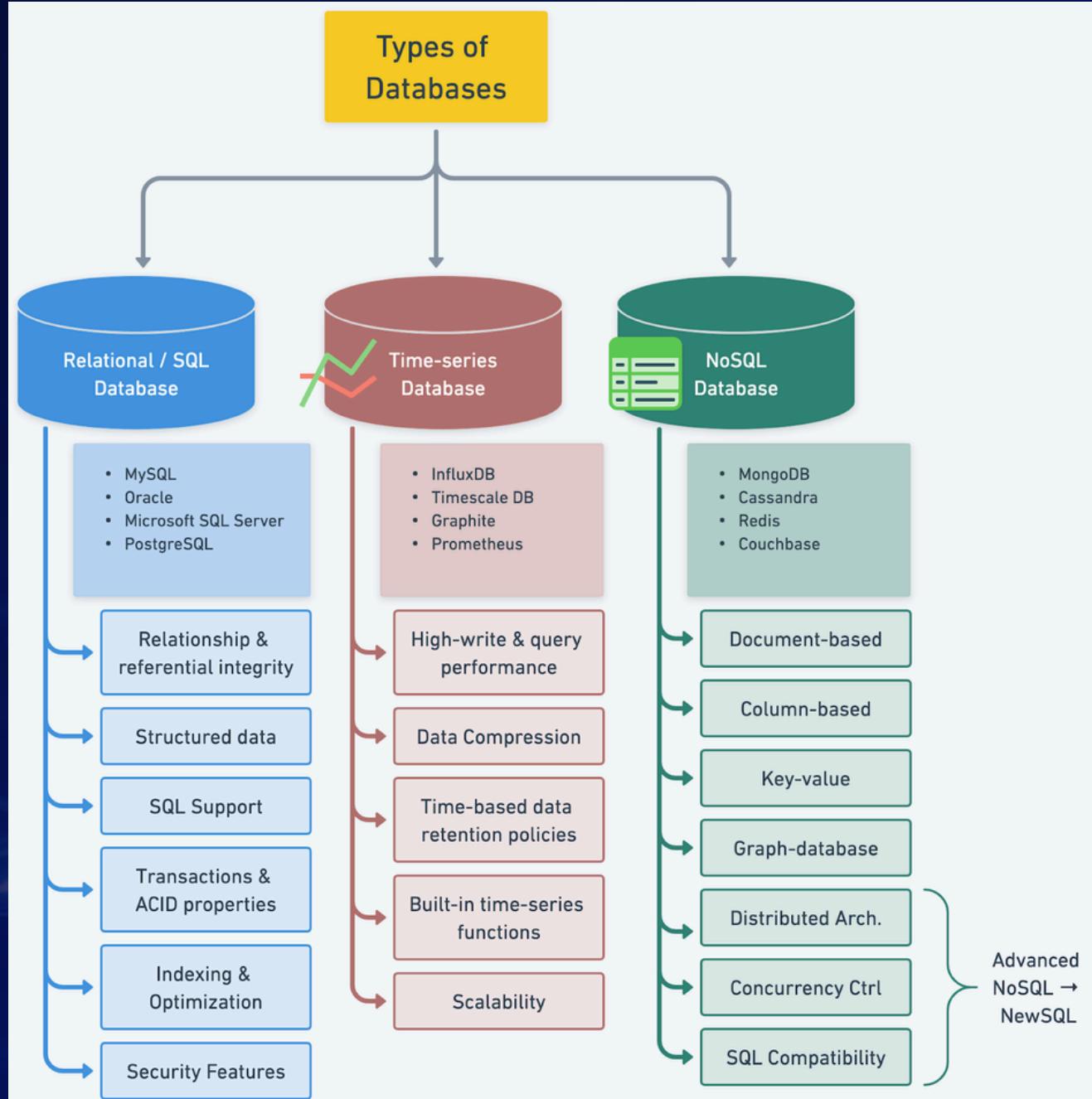
Database testing tasks

Presentation

FATEMA

AREEJ

INTRODUCTION



What is database

Database : is a collection of related data

DBMS : a software that facilitates the creation

Database system =DBMS+database

Database testing

is testing, which is used to analyze the schema, tables triggers, etc., of the database under test. It also assesses data integrity and consistency, which might include creating difficult queries to load and stress test the Database and .review its responsiveness



During the database testing, we can cover the following database activities, such as:

- Testing data integrity
- Checking data validity
- Performance check relate
- Triggers and Functions in the database
- Testing various procedures

What is the purpose of the Database Testing?

Transaction's ACID Properties

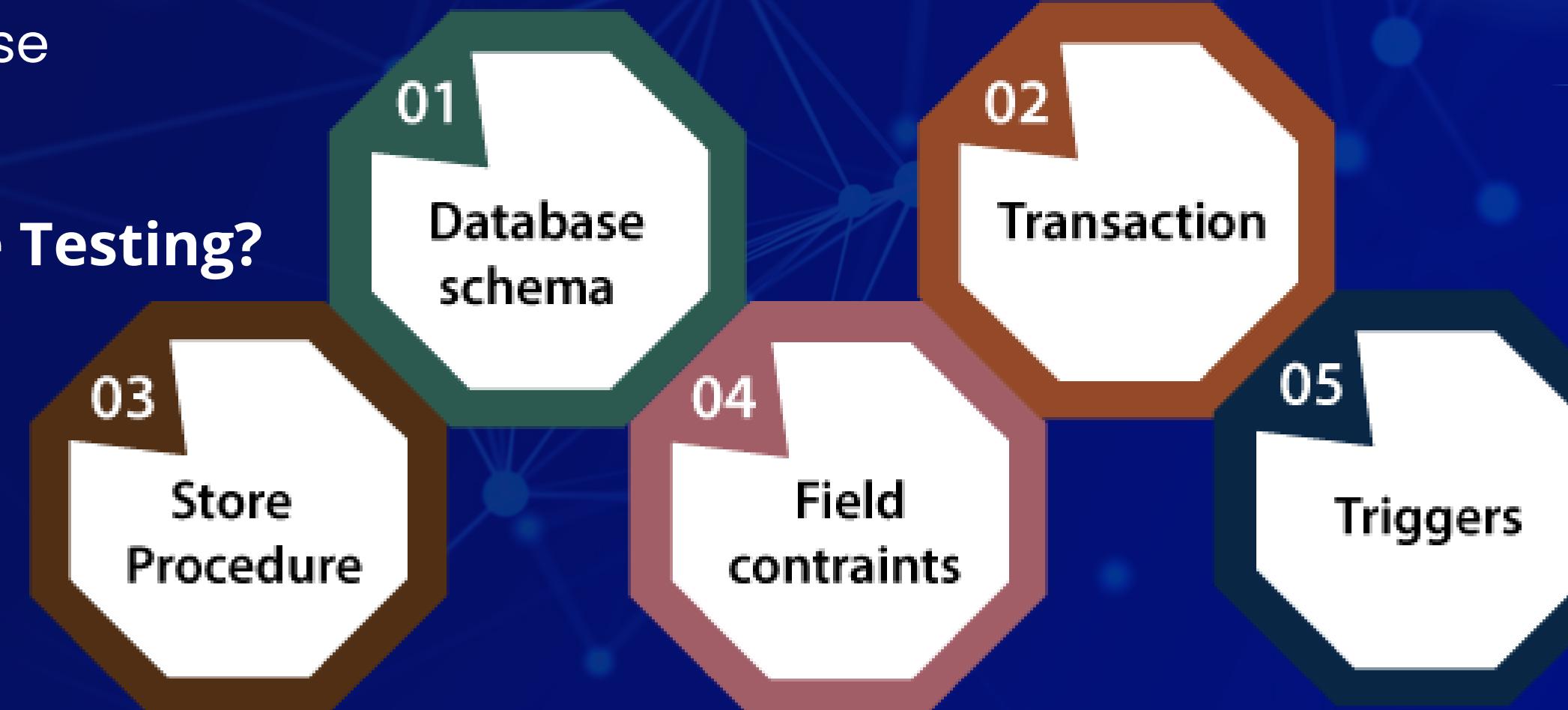
Atomicity
Consistency
Isolation
Durability

Data mapping

Accuracy of Business Rule

Data integrity

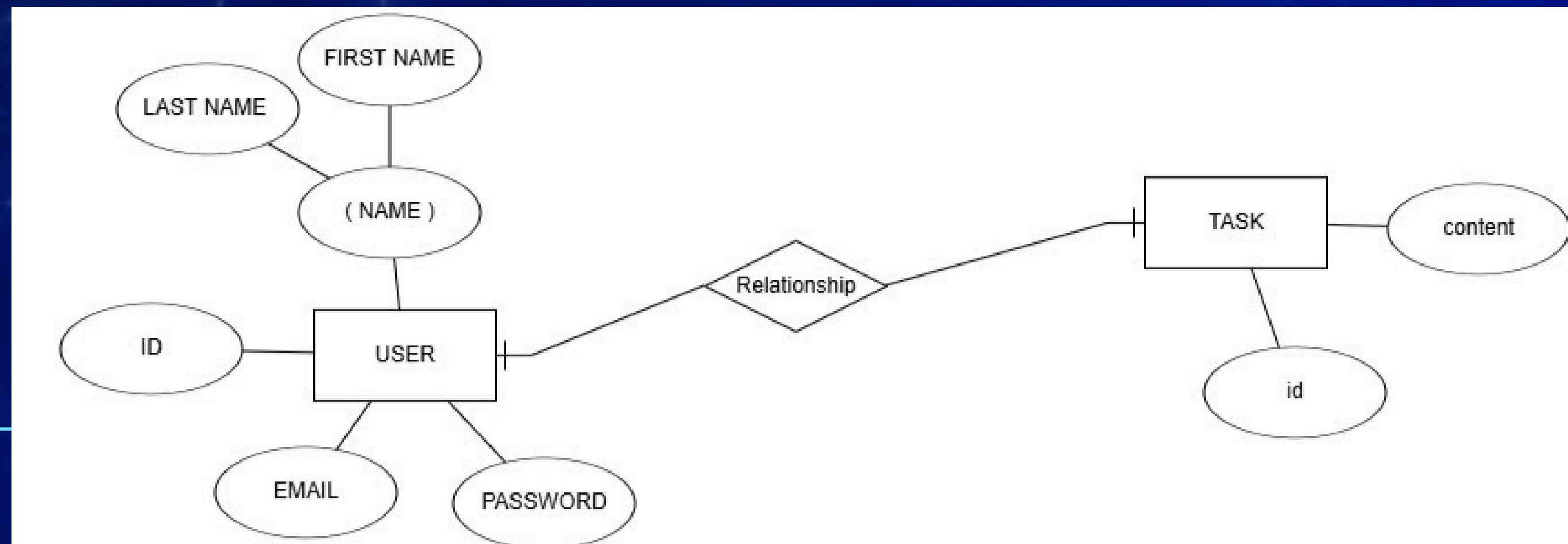
Database Testing Components



ERD

show the logical structure of databases and highlight the interrelations between specific concepts, or entities, (i.e. objects, people, places, etc.) in a given topic area.

First step before we creat database we build ERD as following





Then we open mysql

The screenshot shows the MySQL Workbench interface. In the top-left corner, there are tabs for 'Query 1' and 'SQL File 3'. Below the tabs is a toolbar with various icons. The main area contains two lines of SQL code:

```
1 create database 'final project'  
2
```

In the bottom right corner of the main window, there is a status bar with the text 'Schema: final project' and 'Output' followed by a dropdown menu. Below this, a table titled 'Action Output' shows one entry:

#	Time	Action
1	12:51:16	create database 'final project'

MySQL is an open source RDBMS that uses SQL to create and manage databases

1-we create database called
final project



2-create table users

MySQL80 ×

Query Database Server Tools Scripting Help

Query 1 SQL File 3* users

create database `final project`;

CREATE TABLE `users` (

 `ID` INT PRIMARY KEY AUTO_INCREMENT,

 `first_name` VARCHAR(100) NOT NULL,

 `last_name` VARCHAR(100) NOT NULL,

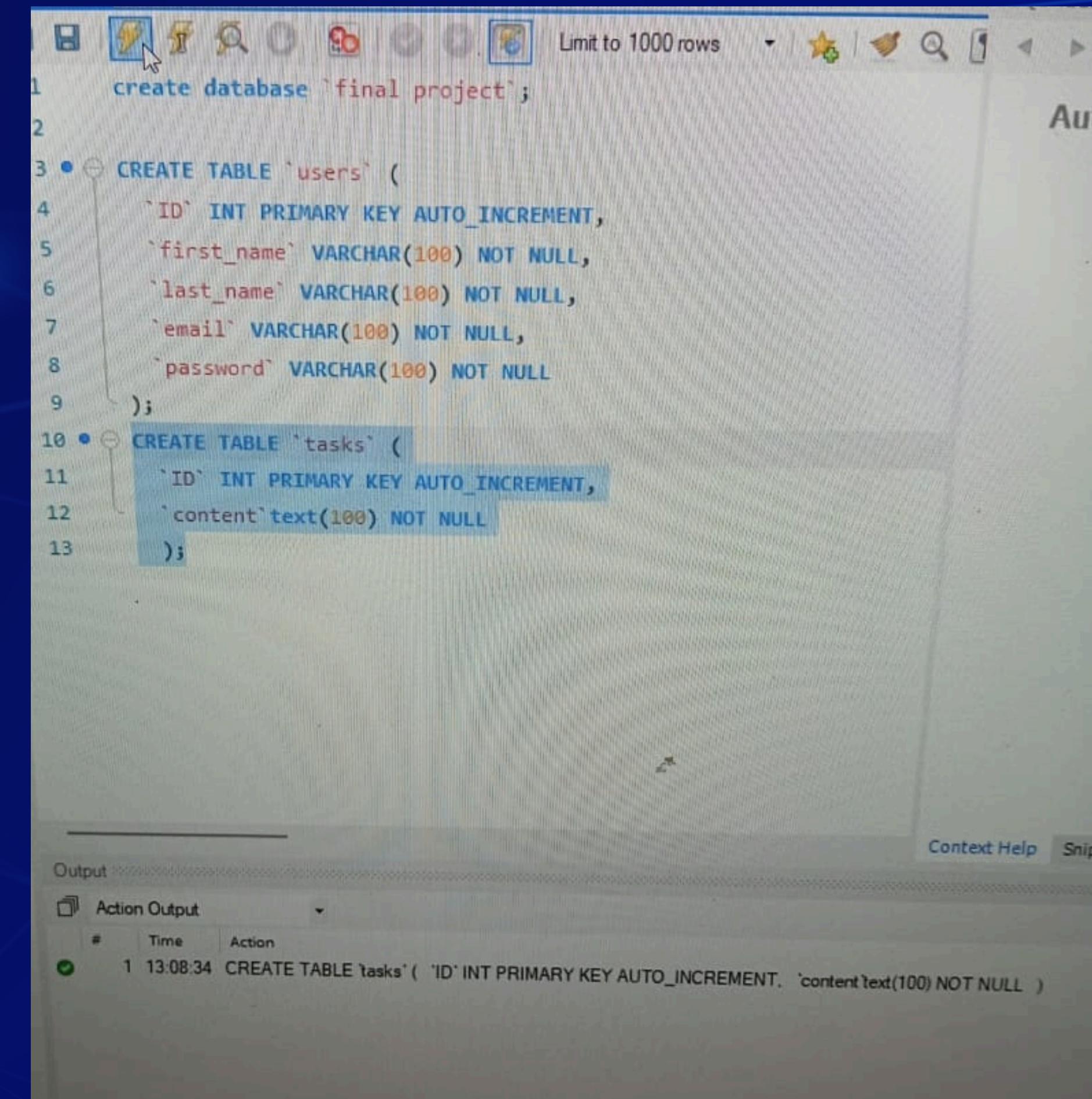
 `email` VARCHAR(100) NOT NULL,

 `password` VARCHAR(100) NOT NULL

);

Context Help Snip...

3-create table tasks



The screenshot shows a MySQL Workbench interface with the following code:

```
1 create database `final project`;
2
3 • CREATE TABLE `users` (
4     `ID` INT PRIMARY KEY AUTO_INCREMENT,
5     `first_name` VARCHAR(100) NOT NULL,
6     `last_name` VARCHAR(100) NOT NULL,
7     `email` VARCHAR(100) NOT NULL,
8     `password` VARCHAR(100) NOT NULL
9 );
10 • CREATE TABLE `tasks` (
11     `ID` INT PRIMARY KEY AUTO_INCREMENT,
12     `content` text(100) NOT NULL
13 );
```

The 'tasks' table definition is highlighted in blue. Below the code, the 'Output' tab shows the execution log:

#	Time	Action
1	13:08:34	CREATE TABLE `tasks`(`ID` INT PRIMARY KEY AUTO_INCREMENT, `content` text(100) NOT NULL)

4-insert a user in table users

first_name :John
last_ name: Doe
Email :john@example.com
Password:ABC@123

The screenshot shows a MySQL database interface with the following details:

- Table Structure:** A table named "users" is shown with columns: id, first_name, last_name, email, and password.
- Query History:** The following SQL code was run:

```
1 • SELECT * FROM `final project`.users;
2 • insert into `users` values
3 (1,'JOHN','DOE','john@example.com', 'ABC@123')
```
- Action Output:** The log shows the execution of the insert query at 13:28:45.

Time	Action
4 13:19:49	insert into `Tasks`(`content`) values ('third task')
5 13:20:06	SELECT * FROM `Final project`.`tasks` LIMIT 0, 1000
6 13:22:51	SELECT `content` FROM `Final project`.`tasks` LIMIT 0, 1000
7 13:24:19	SELECT * FROM `Final project`.`users` LIMIT 0, 1000
8 13:28:45	insert into `users` values (1,'JOHN','DOE','john@example.com', 'ABC@123')
9 13:28:51	SELECT * FROM `Final project`.`users` LIMIT 0, 1000

⌚ **select * from `final project`.
.users**

www.reallygreatsite.com

-

The screenshot shows a MySQL Workbench interface. At the top, there are tabs for 'tasks', 'users', 'users', 'users', and 'users'. Below the tabs, a toolbar has icons for database, table, user, search, and other functions. A message bar at the top right says 'Limit to 1000 rows'. In the main area, a query window contains the SQL command: 'SELECT * FROM `final project`.users;'. Below the query is a 'Result Grid' table with the following data:

ID	first_name	last_name	email	password
1 NULL	JOHN updated	DOE NULL	john@example.com NULL	ABC@123 NULL

At the bottom, a 'users 1' tab is open in the 'Output' section, showing a history of actions:

Time	Action
1 13:13:23	update user 'john' set first_name='JOHN updated', where 'ID'=1
2 13:52:25	SELECT * FROM `final project`.users LIMIT 0, 1000

5-insert in table tasks

The screenshot shows the MySQL Workbench interface with the following details:

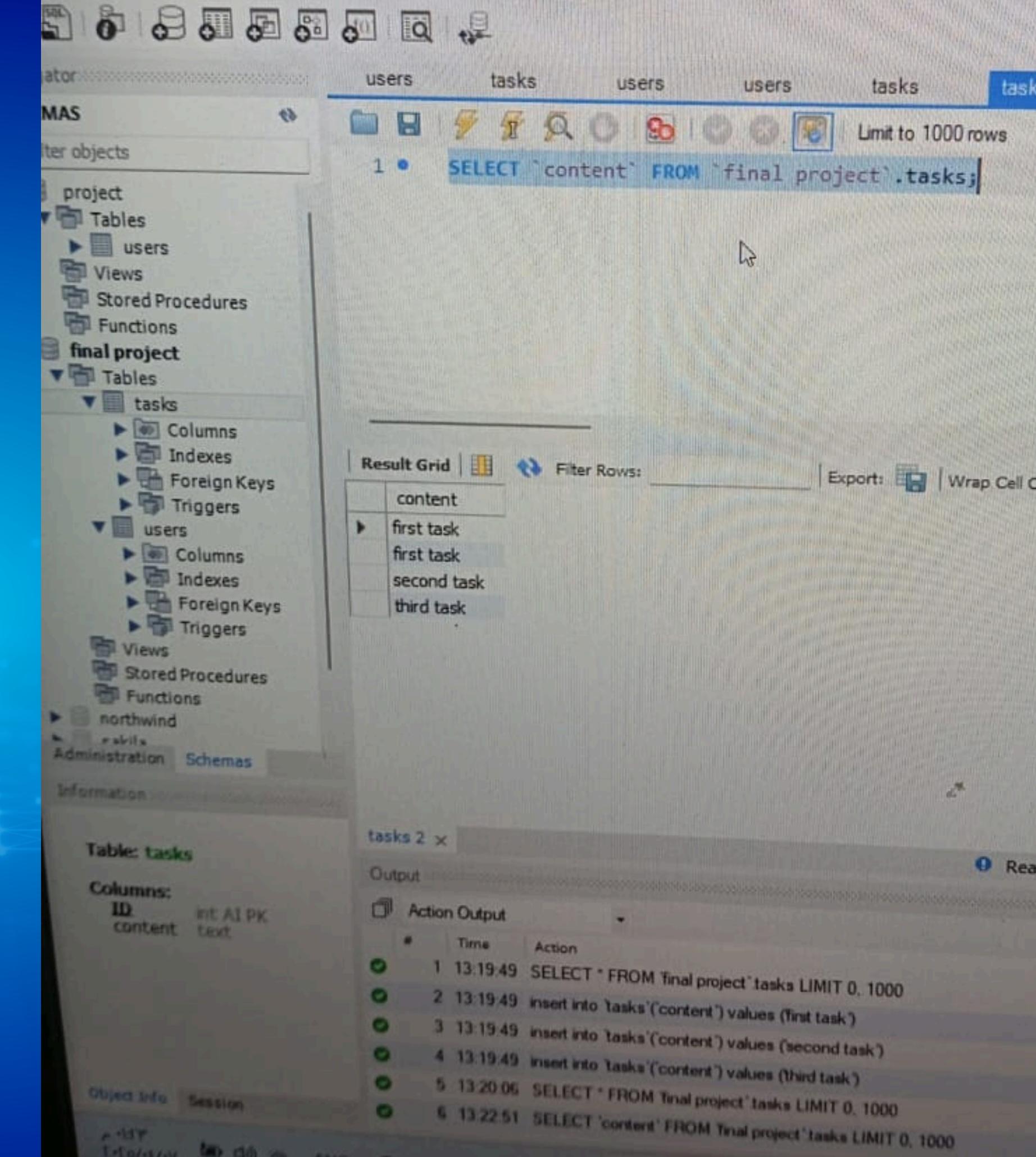
- Toolbar:** Server, Tools, Scripting, Help.
- Connections:** users, tasks, users, users, tasks, tasks.
- SQL Editor:** Contains the following SQL code:

```
1 • SELECT * FROM `final project`.tasks;
2 • insert into `tasks`(`content`) values ('first task');
3 • insert into `tasks`(`content`) values ('second task');
4 • insert into `tasks`(`content`) values ('third task');
```
- Result Grid:** Shows the results of the query:

ID	content
1	first task
2	second task
3	third task
- Action Output:** Displays the log of actions taken:

#	Time	Action	Message
1	13:19:49	SELECT * FROM `final project`.`tasks` LIMIT 0, 1000	1 row(s) returned
2	13:19:49	insert into `tasks`(`content`) values ('first task')	1 row(s) affected
3	13:19:49	insert into `tasks`(`content`) values ('second task')	1 row(s) affected
4	13:19:49	insert into `tasks`(`content`) values ('third task')	1 row(s) affected
5	13:20:06	SELECT * FROM `final project`.`tasks` LIMIT 0, 1000	4 row(s) returned

 select `content` from
'final project'.tasks;



The screenshot shows a MySQL Workbench interface. On the left, the database browser displays a tree structure of databases, tables, and objects. The 'final project' database is selected, and its 'tasks' table is expanded, showing columns like ID and content. In the center, a results grid displays four rows of data from the tasks table, with the first column labeled 'content' and the rows containing 'first task', 'first task', 'second task', and 'third task'. At the bottom, a history window titled 'tasks 2' shows the execution of a SELECT query and three INSERT queries, each inserting a value into the 'content' column of the 'tasks' table.

```
1 • SELECT `content` FROM `final project`.tasks;
```

content
first task
first task
second task
third task

Table: tasks

Columns:

ID	content
	int AI PK text

Output

Action Output

#	Time	Action
1	13:19:49	SELECT * FROM `final project`.`tasks` LIMIT 0, 1000
2	13:19:49	insert into `tasks`(`content`) values ('first task')
3	13:19:49	insert into `tasks`(`content`) values ('second task')
4	13:19:49	insert into `tasks`(`content`) values ('third task')
5	13:20:06	SELECT * FROM `final project`.`tasks` LIMIT 0, 1000
6	13:22:51	SELECT `content` FROM `final project`.`tasks` LIMIT 0, 1000



**update name =john
updated;**

Query 1 tasks users users

Limit to 1000 rows

```
1 •   SELECT * FROM `final project`.users;
2 • update `users`
3     set `first_name`='JOHN updated'
4     where `ID`=1;
5
6
```

Output

Action Output

Time	Action
1 13:51:37	update `users` set `first_name`='JOHN updated' where `ID`=1



DELETE THE USER

www.reallygreatsite.com

The screenshot shows a MySQL Workbench interface. The top navigation bar includes tabs for 'Query 1' (selected), 'tasks', 'users', and 'users'. Below the tabs are various icons for database management. The main area displays two SQL statements:

```
1 • SELECT * FROM `final project`.users;
2 • drop table `users`;
```

The second statement, 'drop table `users`;', is highlighted with a blue selection bar. At the bottom right of the interface, there is a note: 'Limit to 1000 rows'. The bottom section of the window is labeled 'Output' and shows a log entry:

Action Output	Time	Action
1	14:00:15	drop table `users`



2-VALIDATING DATABASE SCHEMA

DESCRIBE users ;

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains four queries:
 - SELECT * FROM `final project`.users;
 - insert into `users` values (1,'JOHN','DOE','john@example.com','ABC@123');
 - (This row is partially visible)
 - DESCRIBE `users`;
- Result Grid:** Displays the schema of the 'users' table.

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	1	auto_increment
first_name	varchar(100)	NO			
last_name	varchar(100)	NO			
email	varchar(100)	NO			
password	varchar(100)	NO			
- Result 2:** Shows the output of the DESCRIBE query.
- Action Output:** Shows the time and action for the DESCRIBE command.



2-VALIDATING DATABASE SCHEMA

show variables like 'auto_increment'

auto_increment_increment – This is the interval between successive AUTO_INCREMENT

auto_increment_offset – This is the starting point for .AUTO_INCREMENT values on that server

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, a code editor window displays a script with several CREATE TABLE statements and a 'show variables like' command. The 'show variables like' command is highlighted in blue. In the bottom right corner of the slide, there's a small preview of the MySQL Workbench results grid.

create database `final project`;

CREATE TABLE `users` (

CREATE TABLE `tasks` (

show variables like 'auto_increment%';

CREATE TABLE users (

);

Result Grid | Filter Rows: Export: Wr

Variable_name	Value
auto_increment_increment	1
auto_increment_offset	1

Result 2 X

Output

Action Output

Time Action

1 19:55:06 DESCRIBE `users`

VALIDATING DATABASE SCHEMA

check data type in id by adding '=' rather than int number

EXPECTED RESULT : ERROR

ACTUAL RESULT :

insert into `users` values
('reyam','ahamed','rey@123.com','rey@123')

Error Code: 1064. You have an error in your SQL syntax; check
the manual that corresponds to your MySQL server version
for the right syntax to use near
('reyam','ahamed','rey@123.com','rey@123') at line 2
0.000 sec

The screenshot shows a MySQL Workbench interface. At the top, a query editor window displays the following SQL code:

```
1 • SELECT * FROM `final project`.users;
2 • insert into `users` values
3 ✘ (=,'reyam','ahamed','rey@123.com',rey@123);
```

The third line has a red error icon next to it. Below the editor is a "Result Grid" table with the following data:

ID	first_name	last_name	email	password
1	John	Doe	john@example.com	ABC@123
2	fff	Doe	john@example.com	ABC@123
3	alaa	ahamed	alaa@example.com	All@123
4	3	ahamed	alaa@example.com	All@123
5	3	ahamed	alaa@example.com	All@123
HULL	HULL	NULL	NULL	HULL

Below the grid is an "Output" pane showing the execution history:

#	Time	Action
1	00:17:31	SELECT * FROM `final project`.users LIMIT 0, 1000
2	00:17:31	insert into `users` values ('reyam','ahamed','rey@123.com',rey@123)



VALIDATING DATABASE SCHEMA

CHECK DUPLICATE DATE

EXPECTED RESULT : ERROR

ACTUAL RESULT :

Error Code: 1062. Duplicate entry
'9' for key 'users.PRIMARY'

base 2 users

Limit to 1000 rows

1 • SELECT * FROM `final project`.users;

2 • insert into `users` values

(9,0,'ahamed','rey@123.com','rey@123');

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid Form Editor Apply Revert Context Help Snippets

ID	first_name	last_name	email	password
1	John	Doe	john@example.com	ABC@123
2	fff	Doe	john@example.com	ABC@123
3	alaa	ahamed	alaa@example.com	All@123
4	3	ahamed	alaa@example.com	All@123
5	3	ahamed	alaa@example.com	All@123
9	10	ahamed	rey@123.com	rey@123
HULL	HULL	HULL	HULL	HULL

Action Output

#	Time	Action
1	00:34:43	SELECT * FROM `final project`.users LIMIT 0, 1000
2	00:34:43	insert into `users` values (9,0,'ahamed','rey@123.com','rey@123')

Message
6 row(s) returned
Error Code: 1062. Duplicate entry '9' for key 'users.PRIMARY'

3

Testing Stored Procedures

Scenario: A database has a stored procedure for user registration. You need to test if it works correctly

The screenshot shows a MySQL Workbench interface. The top window is titled "database 2" and contains the following SQL code:

```
1 •  SELECT * FROM `final project`.users;
2 DELIMITER //
3 •  CREATE PROCEDURE register_user (
4     IN p_first_name VARCHAR(50),
5     IN p_last_name VARCHAR(50),
6     IN p_email VARCHAR(100),
7     IN p_password VARCHAR(100)
8 )
9 BEGIN
10     INSERT INTO users (first_name, last_name, email, password)
11     VALUES (p_first_name, p_last_name, p_email, p_password);
12 END //
13 DELIMITER ;
```

The line "CALL register_user('John','Doe','john@example.com','ABC@123');" is highlighted in blue. Below the code editor is the "Output" pane, which shows the results of the executed command:

Action Output
Time Action
1 01:12:53 CALL register_user('John','Doe','john@example.com','ABC@123')



4 Security Testing: Preventing SQL Injection

Scenario: A web application should prevent SQL injection attacks

.SQL injection is a code injection technique that might destroy your database

.SQL injection is one of the most common web hacking techniques

.SQL injection is the placement of malicious code in SQL statements, via web page input

SQL Injection Based on 1=1 is Always True

A hacker might get access to user names and passwords in a database by

:simply inserting " OR ""="" into the user name or password text box

The screenshot shows a MySQL command-line interface. The command window displays two lines of SQL code:

```
• CALL register_user('John', 'Doe', 'john@example.com', 'ABC@123');
✗ SELECT * FROM users WHERE username = '' OR 1=1; -- AND password = '';
```

The command output pane shows a single log entry:

Action	Time	Action
1	01:30:11	SELECT * FROM users WHERE username = '' OR 1=1 LIMIT 0, 1000

The message pane indicates an error:

Message
Error Code: 1054. Unknown column 'username' in 'where clause'