Task 4.

1. How can one restrict access to certain columns of a database table?

In PostGreSQL the restriction of column done with the GRANT command, it allows to specify types of permissions to certain roles for users or groups on a specific column of a table.   
In example:

**GRANT SELECT (column1, column2) ON table\_name TO username;**

Here, this code grants access to a user called ‘username’ to perform SELECT operations on ‘column1’ and ‘column2’ of table ‘table\_name’  
  
The inverse function of it is the REVOKE when removing the permission. This should be done in the following way:

**REVOKE SELECT (column1, column2) ON table\_name FROM username;**

1. What is the difference between user identification and user authentication?

Both are key components in the access control process.  
**User identification** is where a user claims an identity, usually by providing a unique identifier like a username or an e-mail address. It is basically the act of stating who you are with unique identifiers.

**User authentication** in the other hand is the verification of the claimed identity of the user. The most common method is the password, but nowadays, digital certificate or even biometric authentication such as fingerprint or faceid. So this is a part when you prove that you are the one you claim to be.

1. What are the recommended authentication protocols for PostgreSQL?

* The highly recommended authentications are :
* Password (user use password to log-in)
* Certificate authentication (SSL connection and authentication with SSL certificate)
* GSSAPI Authentication ( It relies on a GSSAPI security library through most commonly Microsoft Active Directory or Kerberos serves)
* LDAP authentication
* RADIUS authentication

1. What is proxy authentication in PostgreSQL and what is it for? Why does it make the previously discussed role-based access control easier to implement?

Proxy authentication in PostgreSQL refers to a mechanism where a user can connect to the database server with one set of credentials – so called proxy user – but then internally their session is switched to another user –target user—for executing queries and accessing database resources. This is particularly useful in scenarios where an intermediary application or service needs to act on behalf of another user in the database without exposing the target user’s credentials.

This approach makes the authentication processes simpler, because it allows the application to manage user sessions and permissions at the database level. By switching roles within a single database connection, the application can enforce access controls based on the authenticated user’s permission WITHOUT needing to maintain separate database connection for each user.