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

One can restrict access by specifying which columns of a table can a certain role or user access via GRANT commands.

For example if one wants a user to be able to view only certain columns of a table:  
  
GRANT SELECT (column1, column2, column3) ON table1 TO username;

To do the same through roles:

CREATE ROLE rolename;

GRANT SELECT (column1, column2, column3) ON table1 TO rolename;

GRANT rolename to username;

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

Identification is the initial step, where the user declares their identity, and provides their username with which they connect to the PostgreSQL database. It is used to identify which user and associated roles are used for the connection, e.g. what the connected user can access.

Authentication is the process of verifying that the connected user is who they claim to be, and that they are indeed authorized to use the permissions granted for the user. There are different methods of authentication, which are configured in the pg\_hba.conf file, which tells PostgreSQL how to authenticate users connecting from different sources.

3. What are the recommended authentication protocols for PostgreSQL?

For most use cases, scram-sha-256 with SSL is recommended. It is a strong password-based authentication method. SSL is used to encrypt communication between the user and the server, so it is protected from third parties.

cert uses SSL certificates to authenticate users, and is recommended in high-security environments, where strong mutual authentication is needed.

peer is ideal for local Unix socket connections.

On top of these authentication protocols, firewalls and role-based access control is always recommended.

4. 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 is when a someone connects using a set of credentials but PostgreSQL executes the session as a different role. It is made possible by tools or extensions, as PostgreSQL doesn’t support proxy authentication natively. It is used for several reasons:

* security: PostgreSQL handles permissions via roles without storing app-specific credentials.
* per user access control: each session operates under the identity of the end user, SQL permissions can apply per user or group.
* auditability: queries can be logged and traced back to the actual user, even if they connected via a shared proxy account

RBAC without proxy authentication requires each user to authenticate directly to PostgreSQL thus one have to manage many user credentials and connections. With proxy authentication users don’t need direct PostgreSQL accounts (or passwords) and the tool or extension used connects using a shared proxy account and activates the appropriate role via SET ROLE. This makes it easier to grant or revoke roles centrally, grant access via groups and control access dynamically.