Screen flow example:

https://trailhead.salesforce.com/content/learn/modules/business_process_automation?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst

Record-triggered flow example:

https://trailhead.salesforce.com/content/learn/modules/record-triggered-flows

Platform Event basics:

https://trailhead.salesforce.com/en/content/learn/modules/platform_events_basics

Salesforce Security:

In Salesforce, the job of ID card is done by Profiles.

Each user has a single profile that controls which objects and features that user has access to.

A profile can be assigned to many users, but a user can have only one profile at a time.

Along with objects, profiles also allow access control to Apps, Tabs, Fields, Page layouts, Record Types, etc.

We don’t typically change a user’s profile once it has been assigned.

We also don’t typically change the configuration of a profile once it has been created.

Let’s say you create a new profile called Recruiter to give recruiters the object-level (security) permissions they need on the Candidate object.

You provide the access to create, read and edit records on the Recruiter profile. CRED - Create, Read, Edit\Update and Delete.

However, granting recruiters these permissions on candidate records does not necessarily mean Recruiters can read or edit every record of the candidate object.

Initially, they can only manage their own records with the help of their profiles.

To manage the data access for the *rest* of the records, for each user, we use something called record level-security.

We control record-level access in **four** ways. These 4 settings together are termed as '*Sharing Settings*'.

They're listed below in the order of increasing access.

1) **Org-wide defaults (OWD) -** specify the default level of access, users have to each other's records.

Org-wide sharing settings are set separately for each type of object.

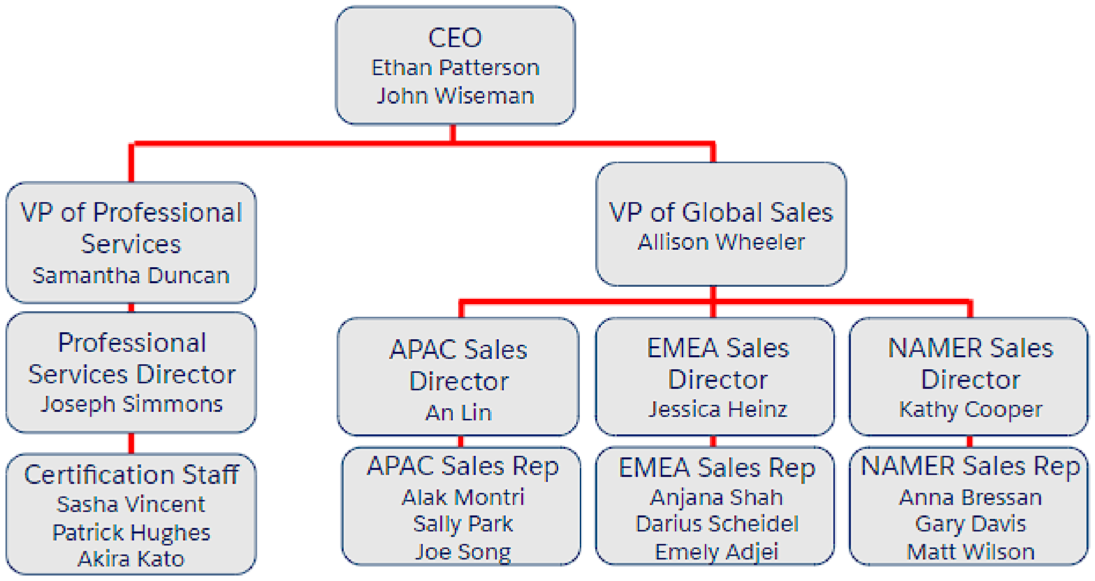
In general, when it comes to Security, we primarily want to make sure that no one has any additional access to any records that they are not supposed to have.

That's why when setting up Salesforce security for any or all objects, first we use org-wide defaults to lock down the data to the most restrictive level possible, and then we use profiles and the other three record-level security tools to grant (open up) access to selected users/groups, **as and when required**.

2) **Role hierarchies-** ensures that managers, along with their own, also have access to the records that are owned by their subordinates.

Hence, users who need to see a lot of data (such as the CEO, top executives, or other management) often appear near the top of the hierarchy.

Roles help us in <u>opening up record access Vertically</u> across the Company Organization.



<u>Profiles</u> are like groups of users that share the same function, e.g. ‘Marketing’, ‘System Admin’, ‘Sales’, and ‘Support’.

<u>Roles</u> are about how users <u>relate</u> to each other in a hierarchy, e.g. the ‘VP of Sales’ is above the ‘Sales Managers’ in the role hierarchy

3) **Sharing rules-** are automatic exceptions to org-wide defaults for particular groups of users, to give them access to records they don’t own or can’t normally see.

If you want to share records automatically with a defined group of users (Public Groups, Roles), Sharing Rules can help you achieve the same.

Sharing Rules are used to <u>extend</u> the Role Hierarchy, so that you are not restricted to the strict top-down sharing as laid out in the role hierarchy – in other words, Sharing Rules can enable you to open up record visibility <u>horizontally</u> across the hierarchy.

4) **Manual sharing-** lets Record owners give read and/or edit permissions <u>manually</u> to users who might not have access to the record any other way.

<u>Note:</u> When the record owner is changed, this record will be removed from the shared access as well.

The 4 types of OWD settings (applied on each object):

Private

Only the record owner, and users above that user’s role in the hierarchy, can view, edit, and report on those records.

Public Read Only

All users can view and report on records, but only the owner, and users above that role in the hierarchy, can edit them.

Public Read/Write

All users can view, edit, and report on all records.

Controlled by Parent

A user can view, edit, or delete a record if she can perform that same action on the (parent) record it belongs to.

Implementing Security for Recruiting App:

OWD settings are used to describe the least level of access a user will require on an object’s records.

So for example, for the ‘Job Posting Site’ object, AW Computing wants OWD to be Public-Read Only i.e. everyone in the company should only be able to view/read all the records by default

And now, AW Computing has asked to put the “Sharing Model” as ‘Private’ for the rest of objects, which means for Candidate, Interview, Job Application, and Position records, only the record owners and their higher management, can view, edit, and report on those records.

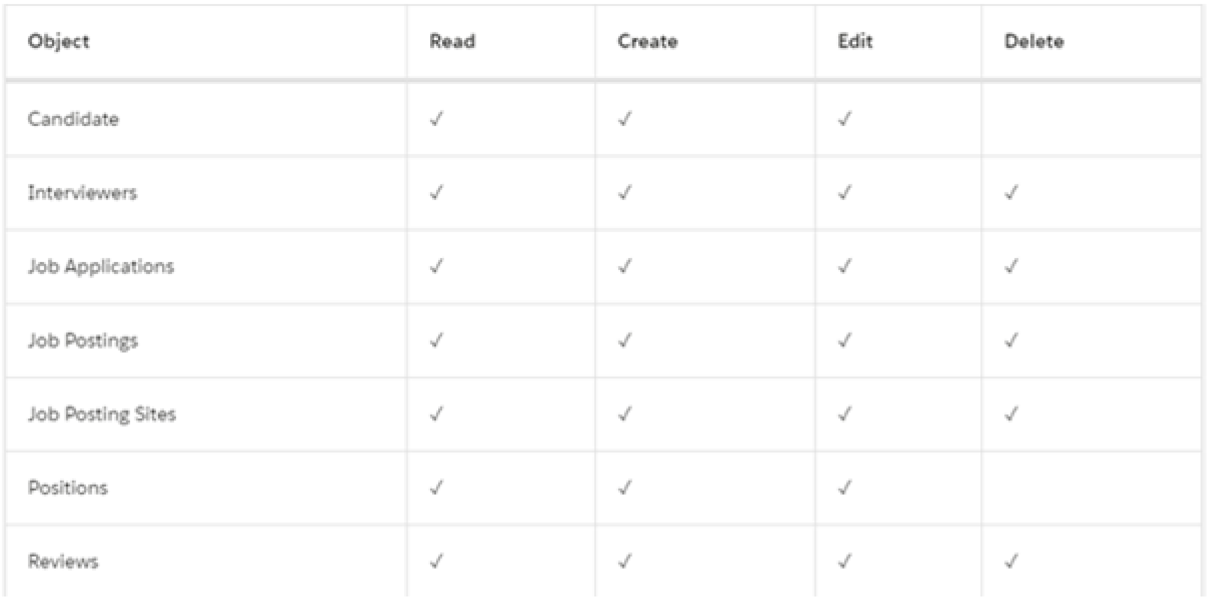
Now it’s time to open the record access to those users who require it.

But before we provide record-access, we first need to provide object-level access, so that users will have the <u>ability</u> to view, edit and delete object instances (records), and so that's why we have to start with profiles.

AW Computing wants only one profile called "HR Recruiter", which will have CRED access to all objects, except for Candidate and Positions.

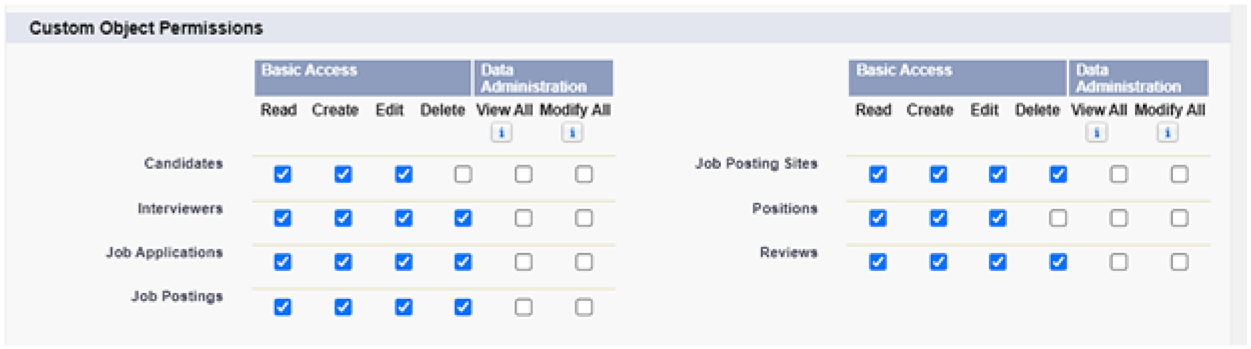
They said "We need to make sure a recruiter will never accidentally delete a record with information about a candidate."

Since we have the profile requirements, let's see it in a tabular format for better understanding when implementing it:



Object	Read	Create	Edit	Delete
Candidate	✓	✓	✓	
Interviewers	✓	✓	✓	✓
Job Applications	✓	✓	✓	✓
Job Postings	✓	✓	✓	✓
Job Posting Sites	✓	✓	✓	✓
Positions	✓	✓	✓	
Reviews	✓	✓	✓	✓

Let's create the profile – HR Recruiter



Custom Object Permissions

	Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All	Modify All
Candidates	✓	✓	✓	☐	☐	☐
Interviewers	✓	✓	✓	✓	☐	☐
Job Applications	✓	✓	✓	✓	☐	☐
Job Postings	✓	✓	✓	✓	☐	☐
Job Posting Sites	✓	✓	✓	✓	☐	☐
Positions	✓	✓	✓	☐	☐	☐
Reviews	✓	✓	✓	✓	☐	☐

Requirement for temporary access:

Create a way to temporarily provide hiring access for those Managers that need to interview candidates for positions in their department. They might take the interview themselves or they might ask a senior member of the team to take the interview.

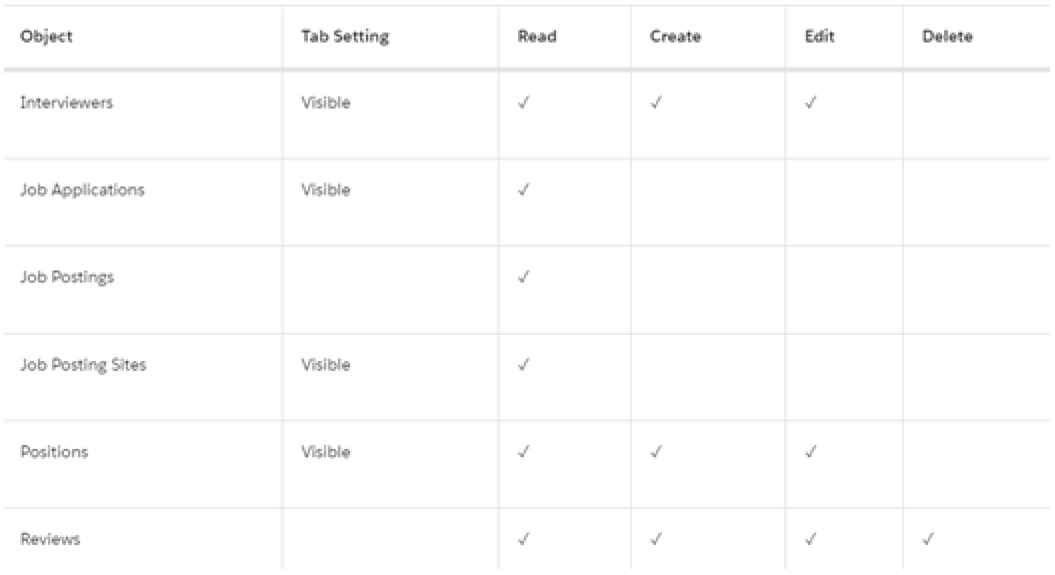
Once the position is filled, the access can be removed.

These types of requirements can be implemented using Permission Sets.

Permission Sets are like Profiles, except that they are used to provide **additional** access to apps, objects, fields (on top of their profiles) for a limited amount of time.

Users can have only ***one*** profile, but they can have multiple permission sets.

Provide access based on this table:



Object	Tab Setting	Read	Create	Edit	Delete
Interviewers	Visible	✓	✓	✓	
Job Applications	Visible	✓			
Job Postings		✓			
Job Posting Sites	Visible	✓			
Positions	Visible	✓	✓	✓	
Reviews		✓	✓	✓	✓

Requirement for managing visibility of specific fields:

The VP of HR has requested that the field ‘Salary Range’ on the Position object be only visible to HR Recruiters, Hiring Managers, and System Admins.

These types of requirements are implemented using Field-Level Security. (FLS)

These are the settings that allow us to protect sensitive fields such as a candidate's social security number without having to hide the candidate object.

You can hide a particular field by removing its access from the Profile.

You can provide access to a particular field by enabling it in Permission Sets.

(Additional) Requirement to share records with HR Recruiters:

For the Recruiting app, we need a sharing rule that shares Job Applications and Reviews created and owned by any member of the org with all recruiters and hiring managers.

For these types of requirements, where we have to share records between two groups of people, it can be implemented using Sharing Rules.

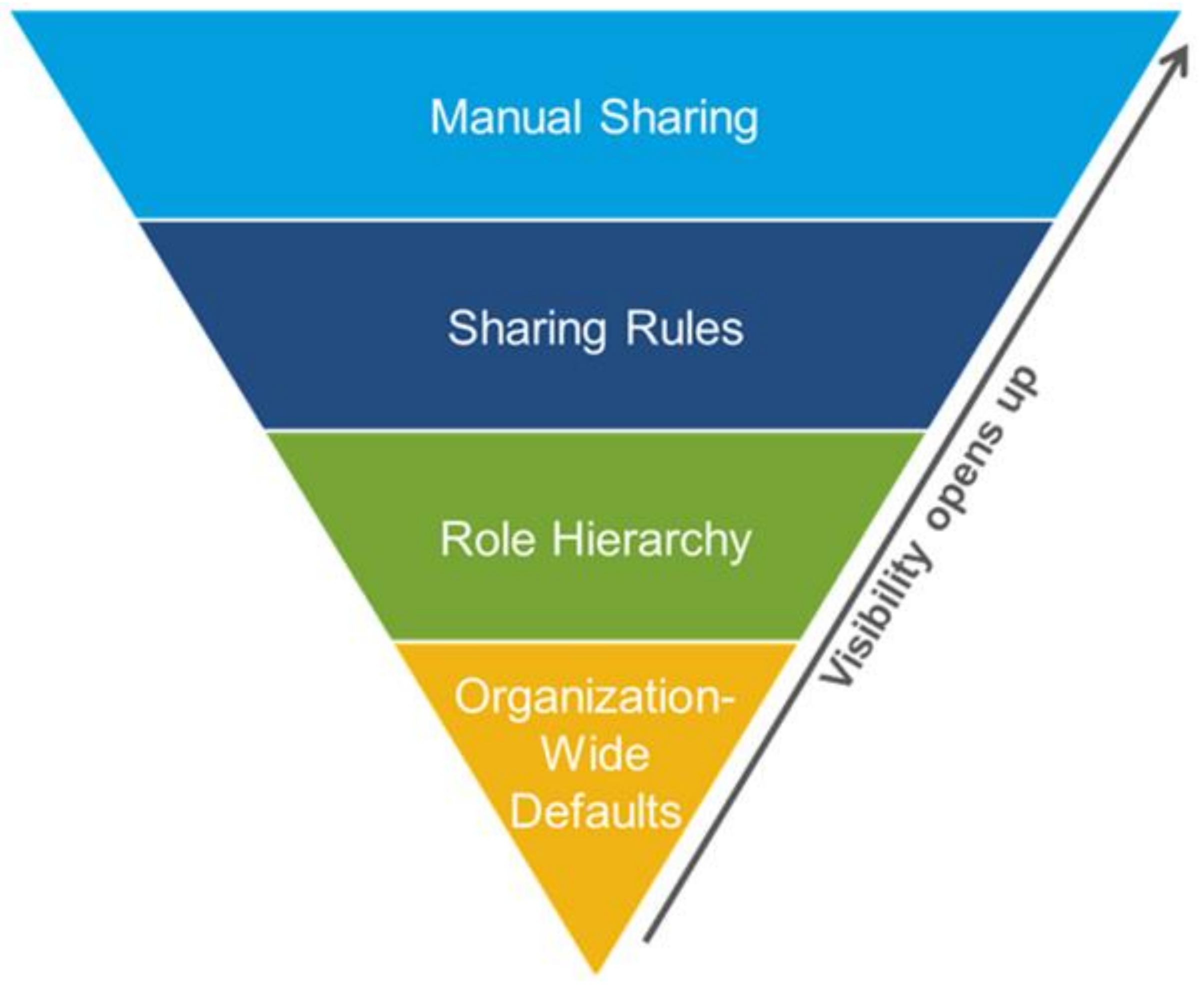
We’ll create a sharing rule that applies to both the Job Application and the Review objects.

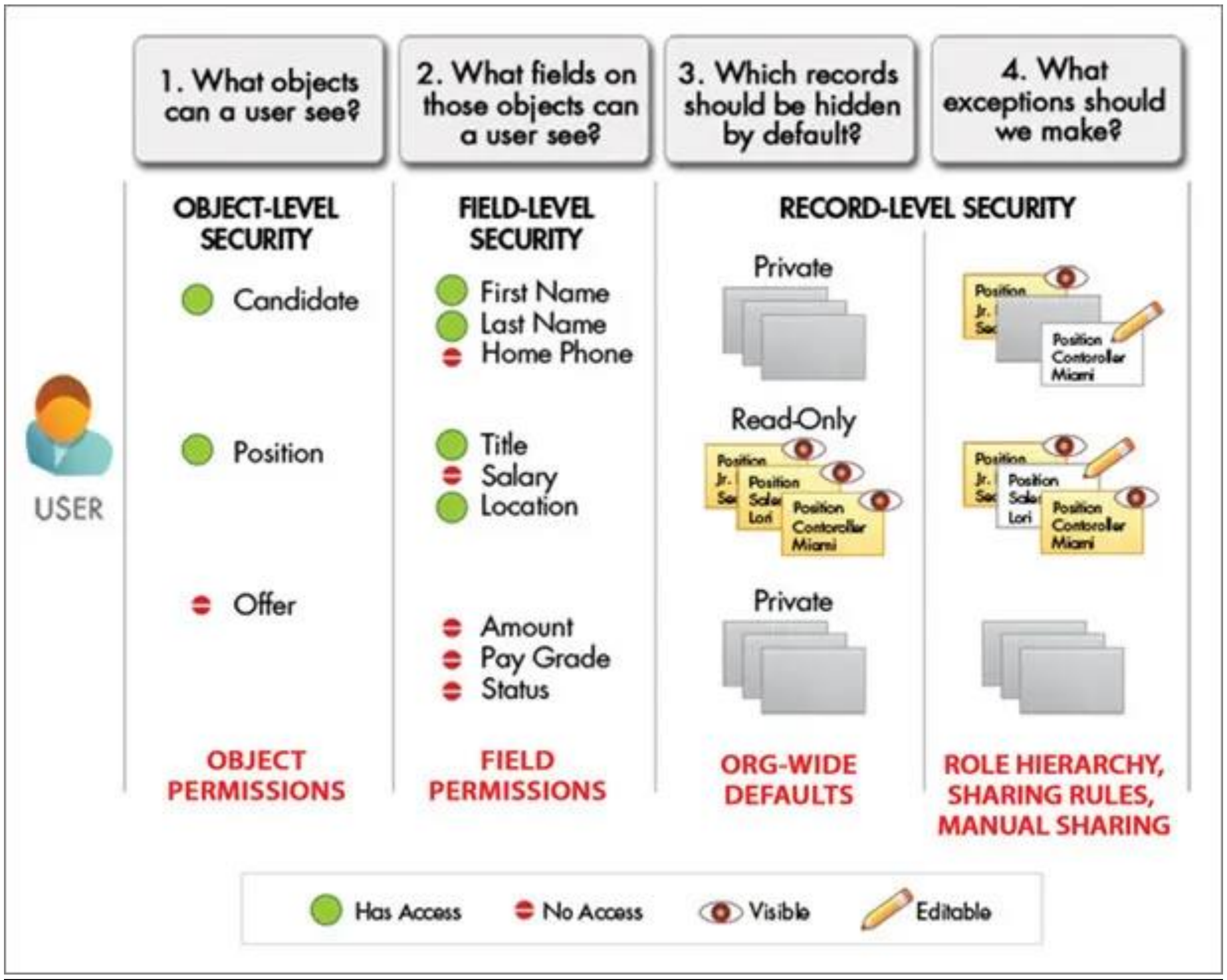
The good news is that we can cover these objects in a single group because the Review object is on the detail side of a master-detail relationship, and so it inherits the sharing settings that we apply to the Job Application object.

Since both recruiters and hiring managers need to read and update access to job applications and reviews, we can define a public group called ‘Reviewers’ that includes both recruiters and hiring managers.

And then we can share all the reviews that are owned by everyone (i.e. ‘All Internal Users’ group) with the ‘Reviewers’ group.

<u>Note</u>: Since we haven’t created the Roles - “Recruiters” and “Hiring Managers”, you can replace them with any other roles such as SVP Human Resources, and VP Marketing




1. What objects can a user see?
2. What fields on those objects can a user see?
3. Which records should be hidden by default?
4. What exceptions should we make?
OBJECT-LEVEL SECURITY
FIELD-LEVEL SECURITY
RECORD-LEVEL SECURITY
USER
Candidate
Position
Offer
First Name
Last Name
Home Phone
Title
Salary
Location
Amount
Pay Grade
Status
Private
Read-Only
Private
Position
Contoroller
Miami
OBJECT PERMISSIONS
FIELD PERMISSIONS
ORG-WIDE DEFAULTS
ROLE HIERARCHY, SHARING RULES, MANUAL SHARING
Has Access
No Access
Visible
Editable

Final Notes:

In Salesforce, we have three layers of Security:

Object-Level Security – Implemented using Profiles and Permission sets

Record-Level Security – Implemented using OWD,

Role Hierarchy, Sharing Rules, and Manual Sharing

Field-Level Security – Implemented using FLS settings on Profiles and Permission sets.

For record-level security, along with the 4 tools mentioned above, we also have other security tools available called "Automated Sharing (Apex\Flow)" and through which we can provide access to records for complex scenarios which are not otherwise possible through Sharing Rules.

For example, Hiring Managers require access to Position records on which they are the Hiring Managers.

Apex Managed Sharing is implemented through Apex code by Salesforce Developers.

Profile access of "View All" or "Modify All" for an object can override all Sharing settings for that particular object.

Profiles deals with Permissions (CRED) (P --> P)

Roles deals with Records (R --> R)

Questions:

1. **How can an App Builder prevent a user from seeing a field in an object?**

    a. Profile
    b. License
    c. Permission Set
    d. Field-Level Security
    e. System Permissions

2. **All Sales users at AW Computing require the same baseline level of permissions to perform their jobs. What's the best way to give this to all of them?**

    a. Profile
    b. License
    c. Permission Set
    d. Field-Level Security
    e. System Permissions

3. **The Sales Managers at AW Computing need additional access that other Sales users do not need. How should this additional access be given?**

   a. Profile
   b. License
   c. Permission Set
   d. Field-Level Security
   e. System Permissions

4. **What feature sets the default level of access users have to records they do not own, in each object?**

   a. Sharing rules
   b. Organization-wide defaults
   c. Role hierarchy
   d. Manual Sharing

5. **At AW Computing, the organization-wide default for Opportunities is**

set to Private. The audit team needs to review all closed Opportunities. How can a app builder meet this requirement?

a. Create an ownership-based sharing rule for Opportunities that gives read/write access to a role.
b. Change the organization-wide default for Opportunities to Public Read/Write.
c. Create a criteria-based sharing rule for Opportunities that gives read access to a public group.
d. Create a new auditor role in the role hierarchy which is above the VP of Sales role.