Q1: What do you notice about the current state of the data?

A.) Main issues

→ Identity/access completeness is not perfect as there are **29** records missing a required access field (Username and/or email). Those people are therefore going to have login/provisioning issues downstream. Certain fields need to therefore be required at account creation

→ Permission alignment is a bigger problem. There are **411** records where the permission flags (teacher/behavior/counselor) do not match what your title-based rules would expect.

→ Titles are free-text and not standardized enough to reliably drive permissions. This makes any rule-based provisioning fragile as small title variations or ambiguous titles can lead to the wrong access.

B.) Trends

→ Over/Under-provisioning risk exists as some roles that do not obviously imply instructional access appear to have instructional permissions set (example: a "Substitute" entry being flagged as a mismatch under the current rules).

→ The dataset looks like it's been maintained without a consistent validation layer as fields are allowed to be empty and permissions are not properly assigned based on the current system.

C.) High Level Trends:

→ Business rules need clarification for edge roles (like substitutes and certain ops/admin roles). The "correct" permissions are partly a policy decision, so the mismatch list is valuable because it surfaces where policy is unclear and helps determine how to approach standardization.

→ There is an **identifier redundancy**, and the dataset would benefit from designating a single authoritative identifier (Ex: Row 26 & 27)

Q2.) How I would approach improving this data

Firstly, understand how data was entered and processed beforehand. Understand how teacher/behavior/counselor permissions are assigned and systems already in place to identify the root of each problem.

Part #A: Cleaning and preparing the data for access to applications

I would start by addressing blocking issues:

→ Identify and resolve records missing required identity fields such as username or email, since these fields are prerequisites for authentication.

→ Generate a list of affected users and route it to the appropriate team for correction rather than attempting to guess or auto-fill sensitive identity data. If data is available then manually enter/correct entries.

Next, I would address permission alignment:

- Validate that Teacher, Behavior, and Counselor permissions align with expectations inferred from job titles.
- For records where permissions do not match expectations, produce an exceptions list rather than immediately overwriting values. Verify and then update where needed.
- Review ambiguous roles (for example, Substitute or Operations roles) with teams to confirm the correct access model before making changes.

Finally, I would normalize the data:

- Standardize job titles to reduce ambiguity and create a standardized title list with matching title codes
- Ensure permission fields are stored consistently (0/1) and changes are documented clearly.

Part B: Creating a process to prevent dirty data in the future
To prevent similar issues from recurring, I would implement validation at the point of entry and before downstream syncs.

- Required-field enforcement: Ensure that critical fields such as username and email must be populated before a staff record can be marked active. If possible, link them to the userID / single unique identifier
- Standardized role selection: Replace free-text job titles with a controlled list of approved roles where possible and attach to a corresponding standardized titlecodes, linked to userID / single unique identifier
- Title-to-permission mapping: Maintain a documented mapping of job titles to expected permissions so access is assigned consistently.
    - Given a userID and their attached title code(s) will allow them to have behavior/teacher/counselor permissions
- Pre-sync validation checks: Run an automated validation step that flags missing identity fields or permission mismatches before records are synced to Clever.

Part C: Building a project plan to implement your process
1.) Audit

- Review existing titles, permissions, and identity fields. Standardize where possible with cross communication with other teams to build criteria/requirements

- Identify high-risk/ambiguous/edge case roles that cause recurring issues. Gather data for missing/inconsistent data that already exist for correction manually or plan to have it reprocessed through the newly built system.

2.) Standardize
- Define approved job titles and associated permissions.
- Document business rules for ambiguous roles (Substitute)

3.) Validate
- Implement automated checks to flag missing identity fields and permission mismatches.
- Produce an exception report rather than silently correcting data and update when approved.

4.) Automate
- Schedule validation to run regularly before system syncs to other platforms.
- Integrate validation into onboarding workflows where possible to ensure required data is present.

5.) Monitor and Iterate
- Track recurring issues and resolution times.
- Update rules as roles or business needs change.

Written Description of Plan and Data Restructure

When dealing with the current data set, the actions I would take would be to identify the data that has missing fields and incorrect permissions based on clarification from the team and newly formed standardization.

To prevent consistent, clean data, ensure required fields are filled at user account creation (username and email, first name and last name) and have each user have a single identifier such as *userID*. This ensures the data will no longer have missing fields that are required.

To address the issue of permissions, create a standardized list of accepted *titles* that can be assigned rather than filled in. Group certain *titles* (such as Teacher, PE Teacher, and edge case titles such as Substitute) into the permission categories available (Teacher/Behavior/Counselor) which will be assigned automatically based on the assigned *title*. Continue to have a *title code* for each standardized *title* and have the standardized title, title code, and permission category assigned to the *userID*.

Technical Approach

I used Python with the pandas library to normalize job titles, derive expected permissions, and identify identity and access issues programmatically. The script flags records with missing access fields and permission mismatches and was used to generate the counts and examples referenced above. This also allowed me to quickly identify data that was incorrect according to the guideline that A.) A username and email is required to have access and B.) Certain Titles get certain permissions.