**In summary:**  
The entire process ensures that the dataset is clean, consistent, and ready for analysis. It fills missing values, removes duplicates, fixes text and date formats, identifies and limits outliers, and prepares numerical data in a standardized way.

1. **Check for missing values**  
   The script first counts how many rows exist in the table and how many of them have missing or empty values in each important column — employee name, salary, hire date, and department. It iterates through each column and every time it finds a Null or empty value, it adds a 1 to the Sum. This helps identify where data is incomplete before cleaning begins.
2. **Handle missing values**  
   Next, it replaces missing or blank text fields such as employee names and departments with the word “Unknown.”  
   For missing salaries, it substitutes the average salary from all other employees.  
   If the hire date is missing, it fills it with a default placeholder date to ensure that every row has a valid value.
3. **Remove duplicate rows**  
   The script removes exact duplicate entries from the table. It keeps only one record from each group of identical rows and deletes the rest. This is done based on all key columns (employee ID, name, salary, hire date, department), ensuring that each employee’s record appears only once. First, we create a subquery, in which we group our records by set of fields with identical values, then choose only one of them with minimum row id, which is unique. Row id is provided by the system. We also keep unique rows, as **minimum row id** is unique too. Using operator not in, we keep all unique rows and delete the rest.
4. **Standardize text columns**  
   To maintain a consistent format, the script trims extra spaces from the beginning and end of text fields and converts all department names to lowercase. This prevents mismatches like “Sales”(capital s) and “sales” (lowercase) from being treated as different departments.
5. **Ensure proper data format**  
   It standardizes the format of the hire date so that all dates follow a consistent “year-month-day” structure. For this we use SUBSTR function. This step removes any extra characters, time components, or formatting inconsistencies.
6. **Detect and handle outliers**  
   The script calculates the minimum, maximum, and average salary values to understand the overall range of salaries. If some salaries are unusually high or low, it caps them within a reasonable range to prevent them from distorting analysis or averages. This SQL snippet creates a temporary dataset (CTE — Common Table Expression) called STATS, which calculates the main statistical indicators for the salary column in the employees table. Three aggregate functions are selected inside the brackets, each of which processes the salary column. After completing this block, we will get a temporary stats table with one row and three columns. Such a CTE can be used, for example:

to output these values (simple statistics);

to apply them in further calculations (for example, to normalize or filter data);

to define the boundaries of "normal" salaries and process outliers.

1. **Normalize numerical data**  
   Finally, it creates a new column *salary\_normalized* that stores normalized salary values, scaled between 0 and 1. This ensures that salaries are comparable on the same scale, which is useful for further analysis, visualization, or machine learning tasks. Here we set what exactly the new value will be equal to. We want to record the normalized salary value calculated using a special formula. This formula is a classic min—max normalization that translates any number into a range from 0 to 1.