Verifying data integrity

Group – 13

IS - 5960: Masters Research Project

Prof. Maria Weber, M.S

20th Feb , 2025.

Members:
1. Srividya Chatakondu
2. Venkata Rahul Chinta
3. Sai Nikhil Reddy Kura
4. Thanusha Reddy Elluru
5. Divya Garapati

Verifying data integrity

Problem Statement:

The Employability Analytics Application will help individuals make better career decisions by analyzing job market trends, skills needed, salaries, and industry growth. It provides tools like interactive career maps, skill assessments, and salary comparisons to guide users effectively. By integrating data from sources like LinkedIn and Glassdoor, it offers personalized recommendations and resources. This project will enhance FutureWorks Solutions reputation, improve service quality, and simplify user career planning. Ultimately, it bridges the gap in the recruitment industry by providing a user-friendly, data-driven platform.

Mapping of Action components to data fields:

Action Component	**Data Fields**
Analyze job market trends	title, location, industry, views
Evaluate necessary skills	skills_desc
Interactive career mapping	title, skills_desc, industry, job_description
Provide skill assessment	skills_desc
Offer personalized recommendations	Job_description, skills_desc, comp_description, speciality

Validation checks:

a. Referential integrity:

Referential integrity is essential to ensure that relationships between different data tables are logically valid. More specifically, it is ensuring that each foreign key within a child table has a related primary key in the parent table.

```
[27] # Filter out job postings with invalid company_ids
     valid_company_ids = df_companies['company_id'].unique()  # Get unique company_ids from companies table
     df_job_posting = df_job_posting[df_job_posting['company_id'].isin(valid_company_ids)]

     # Now the following assertion should pass:
     assert df_job_posting['company_id'].isin(df_companies['company_id']).all(), "Referential integrity failed for company_id in job postings."

     # Check for company_id consistency in industries and specialties tables
     assert df_comp_industries['company_id'].isin(df_companies['company_id']).all(), "Referential integrity issue in industries table."
     assert df_comp_specialties['company_id'].isin(df_companies['company_id']).all(), "Referential integrity issue in specialties table."
```

Process:

Filter Valid IDs: The script then filters the df_job_posting DataFrame to only include those rows for which company_id exists in the df_companies DataFrame. This eliminates orphan records in df_job_posting for which there are no corresponding entries in df_companies.

Assert Validity: Upon filtering, the script uses assert statements to validate that every company_id in df_job_posting, df_comp_industries, and df_comp_specialties has a matching company_id in df_companies. The verification checks for no mismatch or incorrect reference at this stage after the first filter to avoid data inconsistency between tables.

b. Individual field-level integrity:

Field-level integrity involves the activity of ensuring data in every field is accurate, appropriate for the field's data type, and within ranges that are expected. The validation is essential to ensure the validity of derived analyses of data from the dataset.

```
[30] # Validate views are non-negative and realistic
     assert df_job_posting['views'].dropna().apply(lambda x: x >= 0 and x <= 100000).all(), "Invalid values in 'views'."

     # Get unique industries from the DataFrame
     valid_industries = df_comp_industries['industry'].unique().tolist()

     # Check for valid industries (now using all unique industries)
     assert df_comp_industries['industry'].isin(valid_industries).all(), "Invalid industries found."
```

Process:

Validate Numeric Ranges: The program checks that the views column in the df_job_posting DataFrame contains only realistic values (assumed between 0 and 100,000). Initially, dropna() is used to remove any NaN values that might disrupt the validation. The apply() function is then used to force each entry into the specified range.

Validate Categorical Data: Validating the industry field ensures that all entries in the df_comp_industries table are one of the known and acceptable industry categories.
By obtaining a list of valid industries (from the same DataFrame), the script ensures that all entries are proper, which is extremely critical for proper categorization and analysis.

This systematic Python approach ensures that the data utilized in the process of analysis is valid, reliable, and consistent, eliminating errors to an absolute minimum level and rendering any piece of information or conclusion drawn from the data highly reliable.

Manual Operations:

Here, all the operations were performed via code.

Use of Generative AI:

We have used Chat GPT to know about referential integrity and individual field level integrity.

Prompts:

1. Why referential integrity and individual field level integrity is important
2. Generate python code to verify referential integrity and individual field level integrity.