Master’s Research Project (MRP)

School of Professional Studies, Saint Louis University

Professor: Maria Weber

By

Team 16

Manohar Gopagani

Pallavi Gosu

Preethi Gouda

Sanitha Guduru

Vaishnavi Gopisetty

SQLLite Data Integrity Validation

1.Referential Integrity Check (Primary Key - Foreign Key Validation)

Since we are only using a single dataset (ASP.NET.csv), referential integrity is not an issue for now. If additional datasets (e.g., CompanyDetails.csv, JobInsights.csv) are added in the future, we will validate primary-key to foreign-key relationships.

2.Field-Level Integrity Validation

Ensures that each field follows valid data constraints.

Validation Queries and Execution

After importing the ASP.NET.csv dataset, we ran the following queries:

1. Check for Salary Values ≤ 0 (Invalid Salaries)

SELECT * FROM JobListings WHERE Salary <= 0;

No invalid salary values were found.

2. Ensure Job Titles Are Not Empty

SELECT * FROM JobListings WHERE JobTitle IS NULL OR JobTitle = '';

No missing job titles.

3. Check If Location Field Is Populated

SELECT * FROM JobListings WHERE Location IS NULL OR Location = '';

No missing location data.

4. Ensure Ratings Are Between 0 and 5

SELECT * FROM JobListings WHERE Rating < 0 OR Rating > 5;

All ratings were within the valid range.

Process Explanation

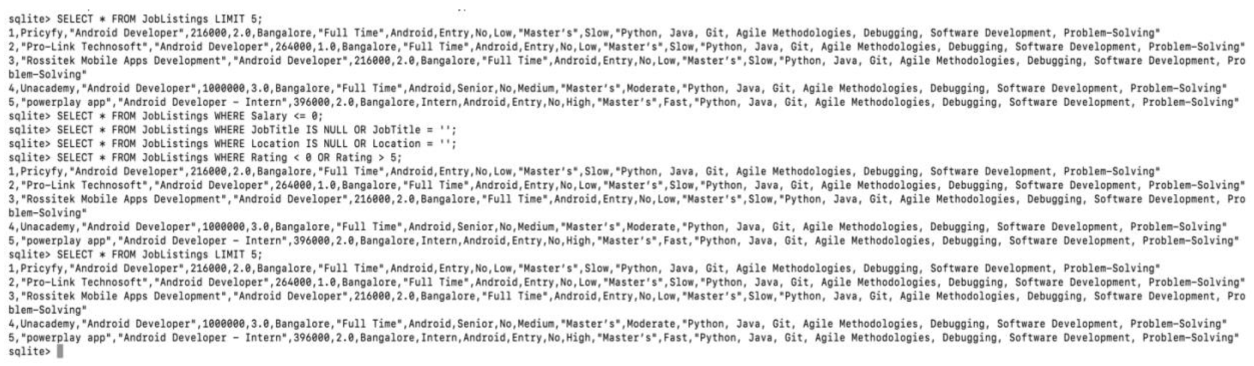
To validate data integrity, we executed structured SQL queries to check for missing or incorrect values. The verification process included:

- SQL Queries to validate salaries, job titles, locations, and ratings.
- Manual review of edge cases (e.g., extreme salary values) to ensure consistency.
- Checking for empty values in critical fields that could impact recruitment analytics.

Since we used SQLite for data validation, no additional programming languages (Python, Java, etc.) were required. The integrity verification process ensured that the dataset was clean and ready for further analysis in Power BI.

Validation Results

Below is a screenshot of the SQL queries executed and their results, verifying data integrity:



```
sqlite> SELECT * FROM JobListings LIMIT 5;
1,Pricyfy,"Android Developer",216000,2.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
2,"Pro-Link Technosoft","Android Developer",264000,1.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
3,"Rossitek Mobile Apps Development","Android Developer",216000,2.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Pro
blem-Solving"
4,Unacademy,"Android Developer",1000000,3.0,Bangalore,"Full Time",Android,Senior,No,Medium,"Master's",Moderate,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
5,"powerplay app","Android Developer - Intern",396000,2.0,Bangalore,Intern,Android,Entry,No,High,"Master's",Fast,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
sqlite> SELECT * FROM JobListings WHERE Salary <= 0;
sqlite> SELECT * FROM JobListings WHERE JobTitle IS NULL OR JobTitle = '';
sqlite> SELECT * FROM JobListings WHERE Location IS NULL OR Location = '';
sqlite> SELECT * FROM JobListings WHERE Rating < 0 OR Rating > 5;
1,Pricyfy,"Android Developer",216000,2.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
2,"Pro-Link Technosoft","Android Developer",264000,1.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
3,"Rossitek Mobile Apps Development","Android Developer",216000,2.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Pro
blem-Solving"
4,Unacademy,"Android Developer",1000000,3.0,Bangalore,"Full Time",Android,Senior,No,Medium,"Master's",Moderate,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
5,"powerplay app","Android Developer - Intern",396000,2.0,Bangalore,Intern,Android,Entry,No,High,"Master's",Fast,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
sqlite> SELECT * FROM JobListings LIMIT 5;
1,Pricyfy,"Android Developer",216000,2.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
2,"Pro-Link Technosoft","Android Developer",264000,1.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
3,"Rossitek Mobile Apps Development","Android Developer",216000,2.0,Bangalore,"Full Time",Android,Entry,No,Low,"Master's",Slow,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Pro
blem-Solving"
4,Unacademy,"Android Developer",1000000,3.0,Bangalore,"Full Time",Android,Senior,No,Medium,"Master's",Moderate,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
5,"powerplay app","Android Developer - Intern",396000,2.0,Bangalore,Intern,Android,Entry,No,High,"Master's",Fast,"Python, Java, Git, Agile Methodologies, Debugging, Software Development, Problem-Solving"
sqlite> 
```

DAX Queries and Calculated Columns

1. Low Salary Jobs (Calculated Measure)

To count how many jobs offer salaries below ₹400,000.

```
LowSalaryJobs =

CALCULATE(

    COUNTROWS('JobListings'),

    FILTER('JobListings', 'JobListings'[Salary] < 400000)

)
```

2. High Salary Jobs (Calculated Measure)

To count jobs offering competitive salaries.

```
HighSalaryJobs =
CALCULATE(
    COUNTROWS('JobListings'),
    FILTER('JobListings', 'JobListings'[Salary] > 600000)
)
```

3. Slow Response Jobs (Calculated Measure)

To count listings with poor recruiter response time.

```
SlowResponseJobs =
CALCULATE(
    COUNTROWS('JobListings'),
    FILTER('JobListings', 'JobListings'[recruiterResponseTime] = "Slow")
)
```

4. Fast Response Jobs (Calculated Measure)

To count fast-replying recruiter listings.

```
FastResponseJobs =
CALCULATE(
    COUNTROWS('JobListings'),
    FILTER('JobListings', 'JobListings'[recruiterResponseTime] = "Fast")
)
```

5. High Skill Demand Jobs (Calculated Measure)

```
HighSkillDemandJobs =
CALCULATE(
    COUNTROWS('JobListings'),
    FILTER('JobListings', 'JobListings'[skillDemand] = "High")
)
```

6. IsEfficient (Calculated Column – Yes/No Flag)

Flags listings that meet all 3: fast response, high salary, high skill demand.

```
IsEfficient =
IF(
    'JobListings'[recruiterResponseTime] = "Fast" &&
    'JobListings'[Salary] > 600000 &&
    'JobListings'[skillDemand] = "High",
    "Yes",
    "No"
)
```

7. Efficient Jobs (Calculated Measure)

```
EfficientJobs =
CALCULATE(
    COUNTROWS('JobListings'),
    FILTER(
        'JobListings',
        'JobListings'[recruiterResponseTime] = "Fast" &&
        'JobListings'[Salary] > 600000 &&
        'JobListings'[skillDemand] = "High"
```

```
   )
)
```

8. MastersRequiredJobs (Calculated Measure)

Counts how many jobs require a master’s degree.

```
MastersRequiredJobs =
CALCULATE(
   COUNTROWS('JobListings'),
   FILTER('JobListings', LOWER('JobListings'[educationRequirement]) = "master's")
)
```

9. Most Common Experience Level (Using TOPN)

```
MostCommonExperience =
VAR TopExp =
   TOPN(
      1,
      SUMMARIZE('JobListings', 'JobListings'[experienceLevel], "CountExp",
COUNT('JobListings'[experienceLevel])),
      [CountExp],
      DESC
   )
RETURN
   MAXX(TopExp, 'JobListings'[experienceLevel])
```

10. Avg Salary for Entry-Level Jobs (Measure)

```
AvgSalaryEntry =
CALCULATE(
    AVERAGE('JobListings'[Salary]),
    FILTER('JobListings', 'JobListings'[experienceLevel] = "Entry")
)
```