**DataSets and Data Cleaning**

**Team Members**

Sowmya Reddy kakularapu

Shivani Kamtam

Sandhya Rani katike

Boni Preetam Kakarla

Yasas Chandra Kakani

Shiva Priya Kasivojjula

**Project Title: SkillBridge**

4rth Semester 2025

**Prof Dr.Maria weber**

Job\_Skills.csv from LinkedIn : <https://www.kaggle.com/datasets/rajatraj0502/linkedin-job-2023>

**1.For Jobs dataset:**

import pandas as pd

import zipfile

from datetime import datetime

import numpy as np

import os

# Step 1: Define local paths

zip\_path = "/Users/sowmyakakularapu/Downloads/job\_skills.csv.zip"

extract\_dir = "/Users/sowmyakakularapu/Downloads"

output\_path = "/Users/sowmyakakularapu/Downloads/Recreated\_Modified\_Job\_Postings.csv"

# Step 2: Extract ZIP and load CSV

with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:

zip\_ref.extractall(extract\_dir)

csv\_file = os.path.join(extract\_dir, "job\_skills.csv")

job\_skills\_df = pd.read\_csv(csv\_file)

# Step 3: Filter for USA jobs

usa\_jobs\_df = job\_skills\_df[job\_skills\_df['Location'].str.contains("United States", na=False)].copy()

# Step 4: Extract City and State

def extract\_city\_state(location):

try:

parts = location.split(',')

if len(parts) >= 3:

return pd.Series([parts[-3].strip(), parts[-2].strip()])

elif len(parts) == 2:

return pd.Series([parts[0].strip(), parts[1].strip()])

else:

return pd.Series(['Unknown', 'Unknown'])

except:

return pd.Series(['Unknown', 'Unknown'])

usa\_jobs\_df[['City', 'State']] = usa\_jobs\_df['Location'].apply(extract\_city\_state)

# Step 5: Combine skills

def combine\_skills(min\_q, pref\_q):

skills = []

if pd.notna(min\_q): skills.append(min\_q.replace('\n', ', '))

if pd.notna(pref\_q): skills.append(pref\_q.replace('\n', ', '))

return ', '.join(skills)

usa\_jobs\_df['RequiredSkills'] = usa\_jobs\_df.apply(

lambda row: combine\_skills(row['Minimum Qualifications'], row['Preferred Qualifications']),

axis=1

)

# Step 6: Simulate Realistic Fields

np.random.seed(42)

sample\_roles = [

'UI/UX Designer', 'Frontend Developer', 'Backend Developer',

'Full Stack Developer', 'Product Manager', 'Data Scientist',

'Machine Learning Engineer', 'DevOps Engineer', 'Cloud Architect',

'Mobile App Developer', 'Software Engineer', 'Site Reliability Engineer',

'AI Research Engineer', 'Cybersecurity Analyst', 'Data Engineer',

'Business Intelligence Analyst', 'Systems Administrator',

'Solutions Architect', 'QA Automation Engineer', 'Technical Program Manager'

]

sample\_companies = [

'Apple', 'Microsoft', 'Google', 'Amazon', 'Meta',

'NVIDIA', 'Intel', 'IBM', 'Salesforce', 'Oracle',

'Adobe', 'Netflix', 'Spotify', 'Zoom', 'Cisco'

]

sample\_salaries = ['$80k-$100k', '$100k-$120k', '$120k-$140k', '$140k-$160k', 'Not Disclosed']

sample\_experience = ['Entry', 'Mid', 'Senior']

sample\_remote = ['Yes', 'No', 'Hybrid']

sample\_jobtype = ['Full-time', 'Internship', 'Contract']

sample\_link = 'https://company.com/job-description'

# Assign synthetic values

usa\_jobs\_df['Role'] = np.random.choice(sample\_roles, size=len(usa\_jobs\_df))

usa\_jobs\_df['JobTitle'] = usa\_jobs\_df['Role']

usa\_jobs\_df['Company'] = np.random.choice(sample\_companies, size=len(usa\_jobs\_df))

usa\_jobs\_df['SalaryRange'] = np.random.choice(sample\_salaries, size=len(usa\_jobs\_df))

usa\_jobs\_df['ExperienceLevel'] = np.random.choice(sample\_experience, size=len(usa\_jobs\_df))

usa\_jobs\_df['RemoteOption'] = np.random.choice(sample\_remote, size=len(usa\_jobs\_df))

usa\_jobs\_df['JobType'] = np.random.choice(sample\_jobtype, size=len(usa\_jobs\_df))

usa\_jobs\_df['Link'] = sample\_link

usa\_jobs\_df['PostedDate'] = pd.to\_datetime(np.random.choice(

pd.date\_range('2025-01-01', '2025-04-25'), size=len(usa\_jobs\_df)))

usa\_jobs\_df['Applications'] = np.random.randint(100, 500, size=len(usa\_jobs\_df))

usa\_jobs\_df['SavedCount'] = np.random.randint(100, 300, size=len(usa\_jobs\_df))

usa\_jobs\_df['ViewCount'] = np.random.randint(200, 1000, size=len(usa\_jobs\_df))

# Step 7: Assemble final structured DataFrame

final\_df = pd.DataFrame()

final\_df['Role'] = usa\_jobs\_df['Role']

final\_df['JobTitle'] = usa\_jobs\_df['JobTitle']

final\_df['Company'] = usa\_jobs\_df['Company']

final\_df['SalaryRange'] = usa\_jobs\_df['SalaryRange']

final\_df['Location'] = usa\_jobs\_df['Location']

final\_df['ExperienceLevel'] = usa\_jobs\_df['ExperienceLevel']

final\_df['RemoteOption'] = usa\_jobs\_df['RemoteOption']

final\_df['JobType'] = usa\_jobs\_df['JobType']

final\_df['Link'] = usa\_jobs\_df['Link']

final\_df['City'] = usa\_jobs\_df['City']

final\_df['State'] = usa\_jobs\_df['State']

final\_df['RequiredSkills'] = usa\_jobs\_df['RequiredSkills']

final\_df['PostedDate'] = usa\_jobs\_df['PostedDate']

final\_df['Applications'] = usa\_jobs\_df['Applications']

final\_df['SavedCount'] = usa\_jobs\_df['SavedCount']

final\_df['ViewCount'] = usa\_jobs\_df['ViewCount']

# Step 8: Export to CSV

final\_df.to\_csv(output\_path, index=False)

print(f"✅ Job dataset saved to: {output\_path}")

**2. USER Dataset**

User data Form:

<https://docs.google.com/forms/d/e/1FAIpQLSfMzRji2kydYBWq63dnbD9I6jaE1DrPezB6ltyatKj0yEABZw/viewform?usp=dialog>

import pandas as pd

import random

# Load your Excel file

input\_path = "/Users/sowmyakakularapu/Downloads/User\_Data.xlsx"

output\_path = "/Users/sowmyakakularapu/Downloads/User\_Data\_With\_RoleSkills.csv"

user\_data\_df = pd.read\_excel(input\_path)

# Define roles, skills, and role descriptions

roles\_skills = {

"Data Scientist": ["Python", "Machine Learning", "Data Analysis", "SQL", "Statistics", "TensorFlow"],

"Data Engineer": ["SQL", "Python", "AWS", "Spark", "Data Warehousing", "ETL"],

"Software Developer": ["Java", "Spring Boot", "SQL", "React", "HTML", "CSS"],

"Cloud Engineer": ["AWS", "Azure", "Linux", "Docker", "Kubernetes", "Terraform"],

"Cybersecurity Analyst": ["Networking", "Python", "Ethical Hacking", "Cybersecurity", "Risk Assessment", "SIEM"]

}

role\_descriptions = {

"Data Scientist": "Analyze and interpret complex data to help companies make decisions.",

"Data Engineer": "Develop, construct, test and maintain architectures such as databases and large-scale processing systems.",

"Software Developer": "Design, build, and maintain software applications.",

"Cloud Engineer": "Design and manage cloud-based infrastructure and services.",

"Cybersecurity Analyst": "Protect systems and networks from cyber threats and attacks."

}

# Add role-specific fields

updated\_users = []

for i, row in user\_data\_df.iterrows():

role = random.choice(list(roles\_skills.keys()))

required\_skills = roles\_skills[role]

known\_skills = random.sample(required\_skills, random.randint(3, 5))

updated\_row = row.copy()

updated\_row["Role"] = role

updated\_row["KnownSkills"] = ", ".join(known\_skills)

updated\_row["RequiredSkills"] = ", ".join(required\_skills)

updated\_row["RoleDescription"] = role\_descriptions[role]

updated\_users.append(updated\_row)

# Save updated DataFrame

final\_users\_df = pd.DataFrame(updated\_users)

final\_users\_df.to\_csv(output\_path, index=False)

print(f"✅ User data updated with roles and saved to:\n{output\_path}")

**3. Job\_Roles\_Growth Dataset**

import pandas as pd

import random

import numpy as np

# File path for your Mac

output\_path = "/Users/sowmyakakularapu/Downloads/Enhanced\_Role\_Growth\_Trends\_Dataset.csv"

# Define roles and associated attributes

roles\_data = {

"Data Scientist": {"Category": "Data", "Industry": "Healthcare"},

"Data Engineer": {"Category": "Data", "Industry": "Finance"},

"Software Developer": {"Category": "Software", "Industry": "Technology"},

"Cloud Engineer": {"Category": "Cloud", "Industry": "Technology"},

"Cybersecurity Analyst": {"Category": "Security", "Industry": "Defense"}

}

years = [2022, 2023, 2024]

seniority\_levels = ["Entry", "Mid", "Senior"]

remote\_options = ["Low", "Medium", "High"]

state\_groups = {

"Data Scientist": "CA, FL, WA",

"Data Engineer": "TX, IL, WA",

"Software Developer": "CA, NY, TX",

"Cloud Engineer": "WA, TX, GA",

"Cybersecurity Analyst": "VA, MD, DC"

}

# Simulate realistic growth percentages and salary by role and year

data = []

for role, attributes in roles\_data.items():

for year in years:

openings = random.randint(1200, 2500)

avg\_salary = random.randint(85000, 160000)

seniority = random.choice(seniority\_levels)

remote = random.choice(remote\_options)

growth = round(np.random.uniform(0, 60), 2)

states = state\_groups[role]

data.append({

"Role": role,

"Year": year,

"Openings": openings,

"Category": attributes["Category"],

"AvgSalary": avg\_salary,

"SeniorityLevel": seniority,

"RemoteAvailability": remote,

"TopHiringStates": states,

"ProjectedGrowth(%)": growth,

"Industry": attributes["Industry"]

})

# Create and save DataFrame

df = pd.DataFrame(data)

df.to\_csv(output\_path, index=False)

print(f"✅ Dataset saved to:\n{output\_path}")

**4. Courses Dataset:**

import pandas as pd

import random

import numpy as np

# Output path

output\_path = "/Users/sowmyakakularapu/Downloads/Courses\_Dataset\_for\_All\_Skills.csv"

# All 99 skills used

skills = [

'Python', 'Pandas', 'TensorFlow', 'Scikit-learn', 'SQL', 'Matplotlib', 'ETL', 'Apache Spark',

'Hadoop', 'Airflow', 'HTML', 'CSS', 'JavaScript', 'React', 'Redux', 'TypeScript', 'Java',

'Spring Boot', 'REST APIs', 'Hibernate', 'PostgreSQL', 'Docker', 'Kubernetes', 'CI/CD',

'Terraform', 'AWS', 'Linux', 'PyTorch', 'MLflow', 'Model Deployment', 'Azure',

'CloudFormation', 'IAM', 'Node.js', 'MongoDB', 'Express.js', 'Kotlin', 'Swift', 'Flutter',

'React Native', 'Xcode', 'Network Security', 'Firewalls', 'SIEM', 'Threat Analysis',

'Encryption', 'Figma', 'Sketch', 'Adobe XD', 'User Research', 'Prototyping', 'Wireframing',

'Selenium', 'Postman', 'TestNG', 'JUnit', 'Automation', 'Manual Testing', 'Deep Learning',

'Neural Networks', 'NLP', 'Computer Vision', 'Unity', 'C#', 'Unreal Engine', 'Blender',

'Game Physics', '3D Modeling', 'Oracle', 'Backup & Recovery', 'Performance Tuning', 'MySQL',

'Excel', 'Power BI', 'Tableau', 'Business Strategy', 'Stakeholder Management', 'Agile',

'Scrum', 'Jira', 'Team Facilitation', 'Sprint Planning', 'Retrospectives', 'Roadmapping',

'User Stories', 'Stakeholder Communication', 'Analytics', 'MVP', 'Solidity', 'Ethereum',

'Smart Contracts', 'Web3.js', 'Crypto Wallets', 'Truffle', 'Monitoring', 'Incident Management',

'SLAs', 'Prometheus', 'Grafana'

]

platforms = {

"Coursera": "https://www.coursera.org/search?query={}",

"Udemy": "https://www.udemy.com/courses/search/?q={}",

"LinkedIn Learning": "https://www.linkedin.com/learning/search?keywords={}"

}

course\_levels = ["Beginner", "Intermediate", "Advanced"]

# Generate rows

rows = []

for skill in skills:

for platform, url\_template in platforms.items():

rows.append({

"Skill": skill,

"CourseName": f"{skill} Mastery with {platform}",

"Platform": platform,

"Link": url\_template.format(skill.lower().replace(" ", "%20")),

"CourseLevel": random.choice(course\_levels),

"DurationHours": random.randint(8, 40),

"Cost": random.choice(["Free", "$29", "$49", "$99"]),

"CertificationAvailable": random.choice(["Yes", "No"]),

"Rating": round(random.uniform(3.5, 5.0), 2),

"ReviewCount": random.randint(500, 100000)

})

# Save to CSV

df = pd.DataFrame(rows)

df.to\_csv(output\_path, index=False)

print(f"✅ Exact dataset regenerated at:\n{output\_path}")

**Derived Tables from User dataset using DAX Formulas**

**1.Skills table**

Skills =

VAR AllSkills =

CONCATENATEX(

'UserData',

SUBSTITUTE([Required Skills], ", ", "|"),

"|"

)

RETURN

DISTINCT(

SELECTCOLUMNS(

ADDCOLUMNS(

GENERATESERIES(1, PATHLENGTH(AllSkills)),

"Skill", PATHITEM(AllSkills, [Value])

),

"Skill", [Skill]

)

)

**2.RoleSkills Table**

RoleSkills =

SELECTCOLUMNS(

GENERATE(

'UserData',

VAR skills = SUBSTITUTE('UserData'[Required Skills], ", ", "|")

RETURN

ADDCOLUMNS(

SELECTCOLUMNS(

GENERATESERIES(1, PATHLENGTH(skills)),

"Skill", PATHITEM(skills, [Value])

),

"Role", 'UserData'[Job Role]

)

),

"Role", [Role],

"Skill", [Skill]

)

**DAX Formulas for Measures**

**1.Match Percent**

MatchPercent =

VAR SelectedSkills = VALUES(Skills[Skill])

VAR CurrentRole = MAX(Roles[Role])

VAR TotalSkills =

CALCULATE(COUNTROWS(RoleSkills), RoleSkills[Role] = CurrentRole)

VAR MatchedSkills =

CALCULATE(

COUNTROWS(RoleSkills),

RoleSkills[Skill] IN SelectedSkills,

RoleSkills[Role] = CurrentRole

)

RETURN

DIVIDE(MatchedSkills, TotalSkills, 1)\*100

**2. TopMatchingRole**

TopMatchingRole =

VAR TopRole =

TOPN(

1,

ADDCOLUMNS(

VALUES(Roles[Role]),

"Score", [MatchPercent]

),

[MatchPercent], DESC

)

RETURN

MAXX(TopRole, Roles[Role])

**3.Missing Skills**

MissingSkills =

VAR SelectedSkills = VALUES(Skills[Skill])

VAR TopRole = [TopMatchingRole]

RETURN

CONCATENATEX(

DISTINCT(

FILTER(

RoleSkills,

RoleSkills[Role] = TopRole &&

NOT RoleSkills[Skill] IN SelectedSkills

)

),

RoleSkills[Skill],

","

)

**4. TopRolematchPercent**

TopRoleMatchPercent =

VAR \_topRole = [TopMatchingRole]

VAR SelectedSkills = VALUES('Skills'[Skill])

VAR TotalRequired =

CALCULATE(

DISTINCTCOUNT('RoleSkills'[Skill]),

'RoleSkills'[Role] = \_topRole

)

VAR MatchedSkills =

CALCULATE(

COUNTROWS(

INTERSECT(

VALUES('RoleSkills'[Skill]),

SelectedSkills

)

),

'RoleSkills'[Role] = \_topRole

)

RETURN

DIVIDE(MatchedSkills, TotalRequired, 0)\*100