Loading Libraries and Data

Final Report

Kelly, Sabrina, Makenzie

October 15, 2025

from pyspark.sql import SparkSession  
import pandas as pd  
import matplotlib.pyplot as plt  
  
raw\_df = pd.read\_csv("data/lightcast\_job\_postings.csv")  
#raw\_df.columns.tolist()

/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_59052/1289692877.py:5: DtypeWarning:  
  
Columns (19,30) have mixed types. Specify dtype option on import or set low\_memory=False.

# Cleaning Data

columns\_to\_drop = [  
 "ID", "URL", "ACTIVE\_URLS", "DUPLICATES", "LAST\_UPDATED\_TIMESTAMP",  
 "NAICS2", "NAICS3", "NAICS4", "NAICS5", "NAICS6",  
 "SOC\_2", "SOC\_3", "SOC\_5"  
]  
raw\_df.drop(columns=columns\_to\_drop, inplace=True)  
  
# Fill missing values  
raw\_df["SALARY"].fillna(raw\_df["SALARY"].median(), inplace=True)  
raw\_df["NAICS\_2022\_6"].fillna("Unknown", inplace=True)  
  
# Drop columns with >50% missing values  
raw\_df.dropna(thresh=len(raw\_df) \* 0.5, axis=1, inplace=True)  
  
raw\_df = raw\_df.drop\_duplicates(subset=["TITLE", "COMPANY", "LOCATION", "POSTED"], keep="first")  
  
#raw\_df.head()

/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_59052/2470702404.py:9: FutureWarning:  
  
A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.  
  
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.  
  
  
  
/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_59052/2470702404.py:10: FutureWarning:  
  
A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.  
  
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.  
  
  
  
/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_59052/2470702404.py:10: FutureWarning:  
  
Setting an item of incompatible dtype is deprecated and will raise an error in a future version of pandas. Value 'Unknown' has dtype incompatible with float64, please explicitly cast to a compatible dtype first.

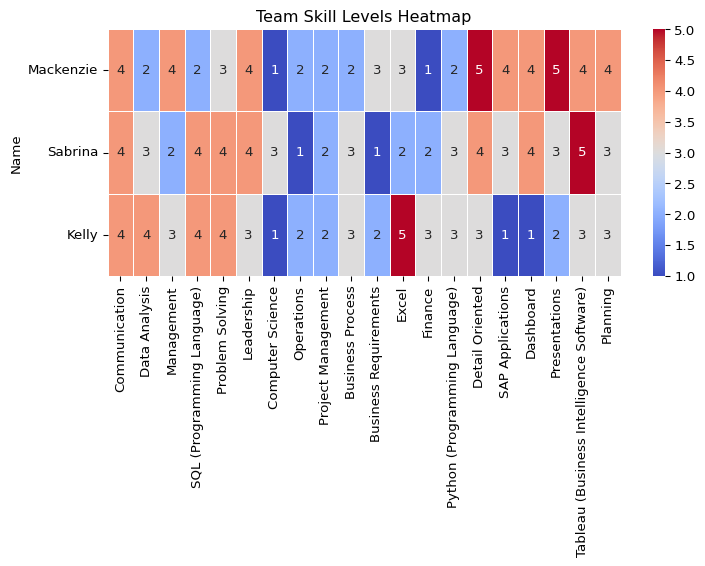
# Create a team-based skill dataframe

Use Scale (1-5) to indicate proficiency levels for each team Member:

* 1 = Beginner
* 2 = Basic knowledge
* 3 = Intermediate
* 4 = Advanced
* 5 = Expert

**Note:** We build the team skill dataframe based on our skills and the most in-demand IT skills and set 1-5 levels for each team member. This allows us to compare side by side in the comparation analysis below.

import pandas as pd  
  
skills\_data = {  
 "Name": ["Mackenzie", "Sabrina", "Kelly"],  
 "Communication": [4, 4, 4],  
 "Data Analysis": [2, 3, 4],  
 "Management": [4, 2, 3],  
 "SQL (Programming Language)": [2, 4, 4],  
 "Problem Solving": [3, 4, 4],  
 "Leadership": [4, 4, 3],  
 "Computer Science": [1, 3, 1],  
 "Operations": [2, 1, 2],  
 "Project Management": [2, 2, 2],  
 "Business Process": [2, 3, 3],  
 "Business Requirements": [3, 1, 2],  
 "Excel": [3, 2, 5],  
 "Finance": [1, 2, 3],  
 "Python (Programming Language)": [2, 3, 3],  
 "Detail Oriented": [5, 4, 3],  
 "SAP Applications": [4, 3, 1],  
 "Dashboard": [4, 4, 1],  
 "Presentations": [5, 3, 2],  
 "Tableau (Business Intelligence Software)": [4, 5, 3],  
 "Planning": [4, 3, 3]  
}  
  
df\_skills = pd.DataFrame(skills\_data)  
df\_skills.set\_index("Name", inplace=True)  
df\_skills  
  
#--- Plot -----  
  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(8, 6))  
sns.heatmap(df\_skills, annot=True, cmap="coolwarm", linewidths=0.5, vmin=1, vmax=5)  
plt.title("Team Skill Levels Heatmap")  
plt.yticks(rotation=0) # keep names horizontal  
plt.tight\_layout()  
plt.savefig("output/team\_skills\_heatmap.png", dpi=300)  
plt.show()  
plt.close()



The team skills levels headmap demostrated we are strogest in communication and problem solving. We also solid in SQL, dashboard /Tableau, and finance. Our weak spots are Computer Science, operations, project management, and writing clear business requiments and process steps.

## Top strengths per person\*\*

top\_strengths = df\_skills.apply(lambda row: row[row == row.max()].index.tolist(), axis=1)  
top\_strengths.to\_frame(name="Top Strength Skills")

|  | Top Strength Skills |
| --- | --- |
| Name |  |
| Mackenzie | [Detail Oriented, Presentations] |
| Sabrina | [Tableau (Business Intelligence Software)] |
| Kelly | [Excel] |

## Team averages by skill

team\_avg = df\_skills.mean().sort\_values(ascending=False)  
team\_avg.to\_frame(name="Team Average (1–5)")

|  | Team Average (1–5) |
| --- | --- |
| Communication | 4.000000 |
| Tableau (Business Intelligence Software) | 4.000000 |
| Detail Oriented | 4.000000 |
| Problem Solving | 3.666667 |
| Leadership | 3.666667 |
| Excel | 3.333333 |
| Presentations | 3.333333 |
| Planning | 3.333333 |
| SQL (Programming Language) | 3.333333 |
| Data Analysis | 3.000000 |
| Dashboard | 3.000000 |
| Management | 3.000000 |
| Business Process | 2.666667 |
| Python (Programming Language) | 2.666667 |
| SAP Applications | 2.666667 |
| Project Management | 2.000000 |
| Finance | 2.000000 |
| Business Requirements | 2.000000 |
| Operations | 1.666667 |
| Computer Science | 1.666667 |

# Compare team skills to industry requirements

## Extract the Most In-Demand Skills from IT Job Postings

from collections import Counter  
import ast  
import pandas as pd  
  
# Parse SKILLS\_NAME into Python lists  
raw\_df["SKILLS\_NAME"] = raw\_df["SKILLS\_NAME"].apply(  
 lambda x: ast.literal\_eval(x) if isinstance(x, str) and x.strip().startswith("[") else (x if isinstance(x, list) else [])  
)  
  
# rename things so similar ones match and stay consistent  
  
alias\_map = {  
 "SQL": "SQL (Programming Language)",  
 "Sql": "SQL (Programming Language)",  
 "MS Excel": "Excel",  
 "Microsoft Excel": "Excel",  
 "PowerBI": "Power BI",  
}  
def canon\_skill(s: str) -> str:  
 s = s.strip()  
 return alias\_map.get(s, s)  
  
# 3) Combines and count  
all\_skills = [canon\_skill(s) for sublist in raw\_df["SKILLS\_NAME"] for s in (sublist if isinstance(sublist, list) else []) if isinstance(s, str)]  
skill\_counts = Counter([s for s in all\_skills if s])  
  
# 4) Top Skills   
top\_skills = [skill for skill, \_ in skill\_counts.most\_common(20)]  
print("Top skills (dataset):", top\_skills)

Top skills (dataset): ['Communication', 'Data Analysis', 'Management', 'SQL (Programming Language)', 'Problem Solving', 'Leadership', 'Computer Science', 'Operations', 'Project Management', 'Business Process', 'Business Requirements', 'Excel', 'Finance', 'Python (Programming Language)', 'Detail Oriented', 'SAP Applications', 'Dashboard', 'Presentations', 'Tableau (Business Intelligence Software)', 'Planning']

## Industry expertise demand (data-driven 1–5)

In this section, We built a simple market target for each top skill using what employers write in job postings. First, we joined the job title and description so we could read the text. If a post listed MIN\_YEARS\_EXPERIENCE, we used it; if not, we pulled numbers like “3+ years” from the text and mapped years to a 1–5 level (with a small bump so “3 years” sits in the middle of a 3–5 range). Next, we read seniority words in the title (junior vs. senior/lead/manager) and skill phrases in the text (basic, intermediate, advanced, expert) and turned those into levels too. For each posting and skill, we combined the three signals, years (50%), seniority (30%), and phrases (20%) to get one score. Finally, we averaged those scores across all postings for each skill to get a 1–5 industry target, which we used to compare with our team’s ratings.

import re  
import numpy as np  
  
# Build text fields if they exist; otherwise empty strings  
title\_col = "TITLE" if "TITLE" in raw\_df.columns else None  
body\_col = "BODY" if "BODY" in raw\_df.columns else None  
  
text\_title = raw\_df[title\_col].astype(str).str.lower() if title\_col else ""  
text\_body = raw\_df[body\_col].astype(str).str.lower() if body\_col else ""  
text\_all = (text\_title + " " + text\_body).astype(str).str.strip()  
  
# Prefer MIN\_YEARS\_EXPERIENCE if present; else parse "3+ years" from text  
min\_col = "MIN\_YEARS\_EXPERIENCE" if "MIN\_YEARS\_EXPERIENCE" in raw\_df.columns else None  
min\_years = pd.to\_numeric(raw\_df[min\_col], errors="coerce") if min\_col else pd.Series(np.nan, index=raw\_df.index)  
  
def extract\_years\_from\_text(text):  
 if not isinstance(text, str): return np.nan  
 m = re.search(r'(\d+)\s\*\+?\s\*(?:years?|yrs)\s+(?:of\s+)?experience', text, flags=re.I)  
 return float(m.group(1)) if m else np.nan  
  
years\_from\_text = text\_all.apply(extract\_years\_from\_text)  
years\_req = min\_years.where(min\_years.notna(), years\_from\_text)  
  
def years\_to\_level\_from\_min(y):  
 if pd.isna(y): return np.nan  
 y = float(y)  
 base = 1 if y < 1 else 2 if y < 2 else 3 if y < 4 else 4 if y < 6 else 5  
 return min(5, base + 0.3) # small bump so "3 years" ≈ mid of typical 3–5  
  
years\_level = years\_req.apply(years\_to\_level\_from\_min)  
  
def seniority\_from\_title(t):  
 if not isinstance(t, str): return np.nan  
 if re.search(r'\b(intern|junior|jr|entry)\b', t): return 2  
 if re.search(r'\b(senior|sr|lead|principal|architect)\b', t): return 5  
 if re.search(r'\b(manager|director|head)\b', t): return 5  
 return 3  
  
seniority\_level = text\_title.apply(seniority\_from\_title) if title\_col else pd.Series(np.nan, index=raw\_df.index)  
  
PHRASE\_LEVELS = [  
 (r'\b(expert|expertise|mastery|guru)\b', 5),  
 (r'\b(advanced|in-depth|strong|proficient|hands-on|solid)\b', 4),  
 (r'\b(intermediate|working knowledge)\b', 3),  
 (r'\b(basic|knowledge of|familiarity)\b', 2),  
]  
def phrase\_level(text):  
 if not isinstance(text, str): return np.nan  
 lvl = np.nan  
 for pat, v in PHRASE\_LEVELS:  
 if re.search(pat, text):  
 lvl = v if pd.isna(lvl) else max(lvl, v)  
 return lvl  
  
phrase\_level\_series = text\_all.apply(phrase\_level)  
  
# Explode one row per (posting, skill), keep only Top 20 skills  
exploded = pd.DataFrame({  
 "SKILLS\_LIST": raw\_df["SKILLS\_NAME"],  
 "years\_level": years\_level,  
 "seniority\_level": seniority\_level,  
 "phrase\_level": phrase\_level\_series  
}).explode("SKILLS\_LIST")  
  
exploded["SKILL"] = exploded["SKILLS\_LIST"].astype(str).apply(canon\_skill)  
exploded = exploded[exploded["SKILL"].isin(top\_skills)]  
  
# Combine signals → expected level per row  
w\_years, w\_seniority, w\_phrase = 0.5, 0.3, 0.2  
def combine\_levels(row):  
 vals, wts = [], []  
 if pd.notna(row["years\_level"]): vals.append(row["years\_level"]); wts.append(w\_years)  
 if pd.notna(row["seniority\_level"]): vals.append(row["seniority\_level"]); wts.append(w\_seniority)  
 if pd.notna(row["phrase\_level"]): vals.append(row["phrase\_level"]); wts.append(w\_phrase)  
 if not vals: return 3.0  
 return float(np.average(vals, weights=wts))  
  
exploded["EXPECTED\_LEVEL"] = exploded.apply(combine\_levels, axis=1)  
  
# Final per-skill target (1–5)  
expected\_per\_skill = (  
 exploded.groupby("SKILL")["EXPECTED\_LEVEL"]  
 .mean()  
 .clip(1,5)  
 .round(2)  
 .reindex(top\_skills)  
)  
expected\_per\_skill.name = "Target (Data-Driven)"  
expected\_per\_skill

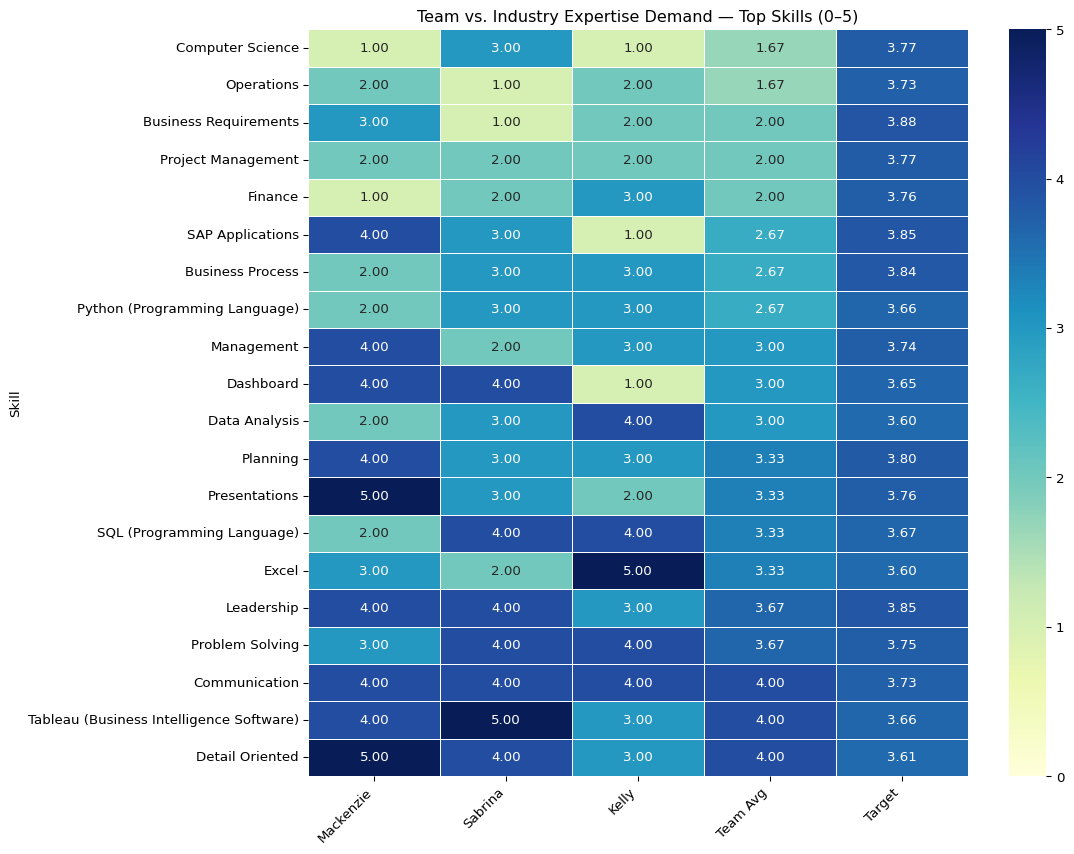
SKILL  
Communication 3.73  
Data Analysis 3.60  
Management 3.74  
SQL (Programming Language) 3.67  
Problem Solving 3.75  
Leadership 3.85  
Computer Science 3.77  
Operations 3.73  
Project Management 3.77  
Business Process 3.84  
Business Requirements 3.88  
Excel 3.60  
Finance 3.76  
Python (Programming Language) 3.66  
Detail Oriented 3.61  
SAP Applications 3.85  
Dashboard 3.65  
Presentations 3.76  
Tableau (Business Intelligence Software) 3.66  
Planning 3.80  
Name: Target (Data-Driven), dtype: float64

## Team Skills Vs. Industry Requiments

import pandas as pd  
import numpy as np  
  
# --- 0) Preconditions: you already have ---  
# df\_skills (index=Name, columns include your top\_skills)  
# top\_skills (list of skills to compare)  
# expected\_per\_skill (Series: index=skills, values=industry target 1–5)  
  
# --- 1) Ensure apples-to-apples (only top\_skills) ---  
df\_team\_top10 = (  
 df\_skills.reindex(columns=top\_skills, fill\_value=0)  
 .apply(pd.to\_numeric, errors="coerce")  
 .fillna(0).clip(0,5)  
)  
  
# --- 2) Build tidy comparison table ---  
team\_avg = df\_team\_top10.mean(axis=0)  
target = expected\_per\_skill.reindex(top\_skills).astype(float)  
gap = (target - team\_avg)  
  
# Order columns: Skill, each person…, Team Avg, Target, Gap  
members = df\_team\_top10.index.tolist()  
rows = []  
for skill in top\_skills:  
 row = {  
 "Skill": skill,  
 \*\*{name: float(df\_team\_top10.loc[name, skill]) for name in members},  
 "Team Avg": round(float(team\_avg[skill]), 2),  
 "Target": round(float(target[skill]), 2) if pd.notna(target[skill]) else np.nan,  
 "Gap (Target−Avg)": round(float(gap[skill]), 2) if pd.notna(gap[skill]) else np.nan,  
 }  
 rows.append(row)  
  
comparison\_table = pd.DataFrame(rows)  
  
# Sort by biggest gap (needs first)  
comparison\_table = comparison\_table.sort\_values("Gap (Target−Avg)", ascending=False, na\_position="last").reset\_index(drop=True)  
  
# Nice rounding for member columns too  
for name in members:  
 comparison\_table[name] = comparison\_table[name].round(2)  
  
comparison\_table

|  | Skill | Mackenzie | Sabrina | Kelly | Team Avg | Target | Gap (Target−Avg) |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Computer Science | 1.0 | 3.0 | 1.0 | 1.67 | 3.77 | 2.10 |
| 1 | Operations | 2.0 | 1.0 | 2.0 | 1.67 | 3.73 | 2.06 |
| 2 | Business Requirements | 3.0 | 1.0 | 2.0 | 2.00 | 3.88 | 1.88 |
| 3 | Project Management | 2.0 | 2.0 | 2.0 | 2.00 | 3.77 | 1.77 |
| 4 | Finance | 1.0 | 2.0 | 3.0 | 2.00 | 3.76 | 1.76 |
| 5 | SAP Applications | 4.0 | 3.0 | 1.0 | 2.67 | 3.85 | 1.18 |
| 6 | Business Process | 2.0 | 3.0 | 3.0 | 2.67 | 3.84 | 1.17 |
| 7 | Python (Programming Language) | 2.0 | 3.0 | 3.0 | 2.67 | 3.66 | 0.99 |
| 8 | Management | 4.0 | 2.0 | 3.0 | 3.00 | 3.74 | 0.74 |
| 9 | Dashboard | 4.0 | 4.0 | 1.0 | 3.00 | 3.65 | 0.65 |
| 10 | Data Analysis | 2.0 | 3.0 | 4.0 | 3.00 | 3.60 | 0.60 |
| 11 | Planning | 4.0 | 3.0 | 3.0 | 3.33 | 3.80 | 0.47 |
| 12 | Presentations | 5.0 | 3.0 | 2.0 | 3.33 | 3.76 | 0.43 |
| 13 | SQL (Programming Language) | 2.0 | 4.0 | 4.0 | 3.33 | 3.67 | 0.34 |
| 14 | Excel | 3.0 | 2.0 | 5.0 | 3.33 | 3.60 | 0.27 |
| 15 | Leadership | 4.0 | 4.0 | 3.0 | 3.67 | 3.85 | 0.18 |
| 16 | Problem Solving | 3.0 | 4.0 | 4.0 | 3.67 | 3.75 | 0.08 |
| 17 | Communication | 4.0 | 4.0 | 4.0 | 4.00 | 3.73 | -0.27 |
| 18 | Tableau (Business Intelligence Software) | 4.0 | 5.0 | 3.0 | 4.00 | 3.66 | -0.34 |
| 19 | Detail Oriented | 5.0 | 4.0 | 3.0 | 4.00 | 3.61 | -0.39 |

import os  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Pick which columns to plot (only numbers!)  
members = df\_team\_top10.index.tolist()   
numeric\_cols = members + ["Team Avg", "Target"]   
  
# Make a numeric matrix indexed by Skill  
heatmap\_df = (  
 comparison\_table  
 .set\_index("Skill")[numeric\_cols]  
 .apply(pd.to\_numeric, errors="coerce")  
)  
  
# 3) Plot  
os.makedirs("output", exist\_ok=True)  
plt.figure(figsize=(12, max(6, 0.45\*len(heatmap\_df)))) # grow height if many skills  
sns.heatmap(  
 heatmap\_df, annot=True, fmt=".2f",  
 cmap="YlGnBu", linewidths=0.5, vmin=0, vmax=5  
)  
plt.title("Team vs. Industry Expertise Demand — Top Skills (0–5)")  
plt.xticks(rotation=45, ha="right")  
plt.yticks(rotation=0)  
plt.tight\_layout()  
plt.savefig("output/team\_vs\_industry\_expertise\_heatmap.png", dpi=300)  
plt.show()  
plt.close()



The team vs. industry headmad compares our skills to the market targets (bottom row). We meet or beat the target in communication and Tableau and are close on problem solving, Excel, and detail-oriented work. We are below the target in computer science, operations, project management, business requirements/process, leadership, finance, Python, and SQL. Our focus should be to raise those areas to about 3.5 or 4 with short courses and practice. Overall, we explain and show insights well, but we need stronger basics and delivery skills to match the market.

## Gaps + Market Adjusted Priorities

# Unweighted gap (Target − Team Avg)  
team\_avg = df\_team\_top10.mean()  
gap = expected\_per\_skill - team\_avg  
  
gap\_table = (  
 pd.DataFrame({  
 "Avg Team Level": team\_avg.round(2),  
 "Target (Data-Driven)": expected\_per\_skill.round(2),  
 "Gap (Target − Avg)": gap.round(2)  
 })  
 .sort\_values("Gap (Target − Avg)", ascending=False)  
)  
gap\_table

|  | Avg Team Level | Target (Data-Driven) | Gap (Target − Avg) |
| --- | --- | --- | --- |
| Computer Science | 1.67 | 3.77 | 2.10 |
| Operations | 1.67 | 3.73 | 2.06 |
| Business Requirements | 2.00 | 3.88 | 1.88 |
| Project Management | 2.00 | 3.77 | 1.77 |
| Finance | 2.00 | 3.76 | 1.76 |
| SAP Applications | 2.67 | 3.85 | 1.18 |
| Business Process | 2.67 | 3.84 | 1.17 |
| Python (Programming Language) | 2.67 | 3.66 | 0.99 |
| Management | 3.00 | 3.74 | 0.74 |
| Dashboard | 3.00 | 3.65 | 0.65 |
| Data Analysis | 3.00 | 3.60 | 0.60 |
| Planning | 3.33 | 3.80 | 0.47 |
| Presentations | 3.33 | 3.76 | 0.43 |
| SQL (Programming Language) | 3.33 | 3.67 | 0.34 |
| Excel | 3.33 | 3.60 | 0.27 |
| Leadership | 3.67 | 3.85 | 0.18 |
| Problem Solving | 3.67 | 3.75 | 0.08 |
| Communication | 4.00 | 3.73 | -0.27 |
| Tableau (Business Intelligence Software) | 4.00 | 3.66 | -0.34 |
| Detail Oriented | 4.00 | 3.61 | -0.39 |

The gaps table shows where we sit below the market target. Positive numbers mean we need to improve; negative numbers mean we already meet or beat the target. Our biggest gaps are in computer science, operations, business requirements, project management, and finance. Smaller gaps show up in SAP, business process, and Python. We already match or exceed the market in communication, Tableau, and being detail-oriented.

To set priorities, we adjust each gap by how common the skill is in job postings (market-adjusted priority). Using that, we should first focus on computer science and operations, then business requirements and project management, followed by finance/SAP. This plan lets us close the largest, most market-relevant gaps first while keeping our strengths sharp.

# Improvement Plan

To improve our IT skills, we can focus on computer science basics, business requirements, project management, and operations. Mackenzie can prioritize computer science, then operations, finance, and project management. Sabrina will focus on project management, business requirements, and operations. Finally, Kelly can focus on computer science, then business requirements, operations, and SAP basics. We can keep our strengths, such as communication, Tableau, SQL, and problem-solving, refreshed with light weekly practice.

Additionally, we could use the following resources to improve our skills: Harvard CS50x for an introduction to computer science (a free course) and LeetCode for practice. For the programming languages SQL and Python, we can use Coursera and DataCamp courses, and LeetCode for programming exercises, which are really helpful for interview practice. For Project Management and operations, we can take the Google Project Management on Coursera. For business requirements, we can take the Managing Requirements for the Business Analysis course on Pluralsight. For finance, we can use Khan Academy’s accounting and financial statements modules. For SAP, we can use SAP Learning’s free fundamentals. For Excel, we can follow Microsoft Learn’s Excel path to reach level 3+.

Finally, we can address our skill gaps with a simple schedule. We can meet weekly or every other week and rotate knowledge guides for each session. If someone is great at project management or another skill, they will share their knowledge and tips. We will work on small, practical projects where we code together, review each other’s work, and discuss what we find. We can keep a shared document and a simple dashboard using a template to track each person’s goals, progress, and next steps for each area that needs improvement. After each session, we will note what we learned, what held us back, and what went well to keep improving.