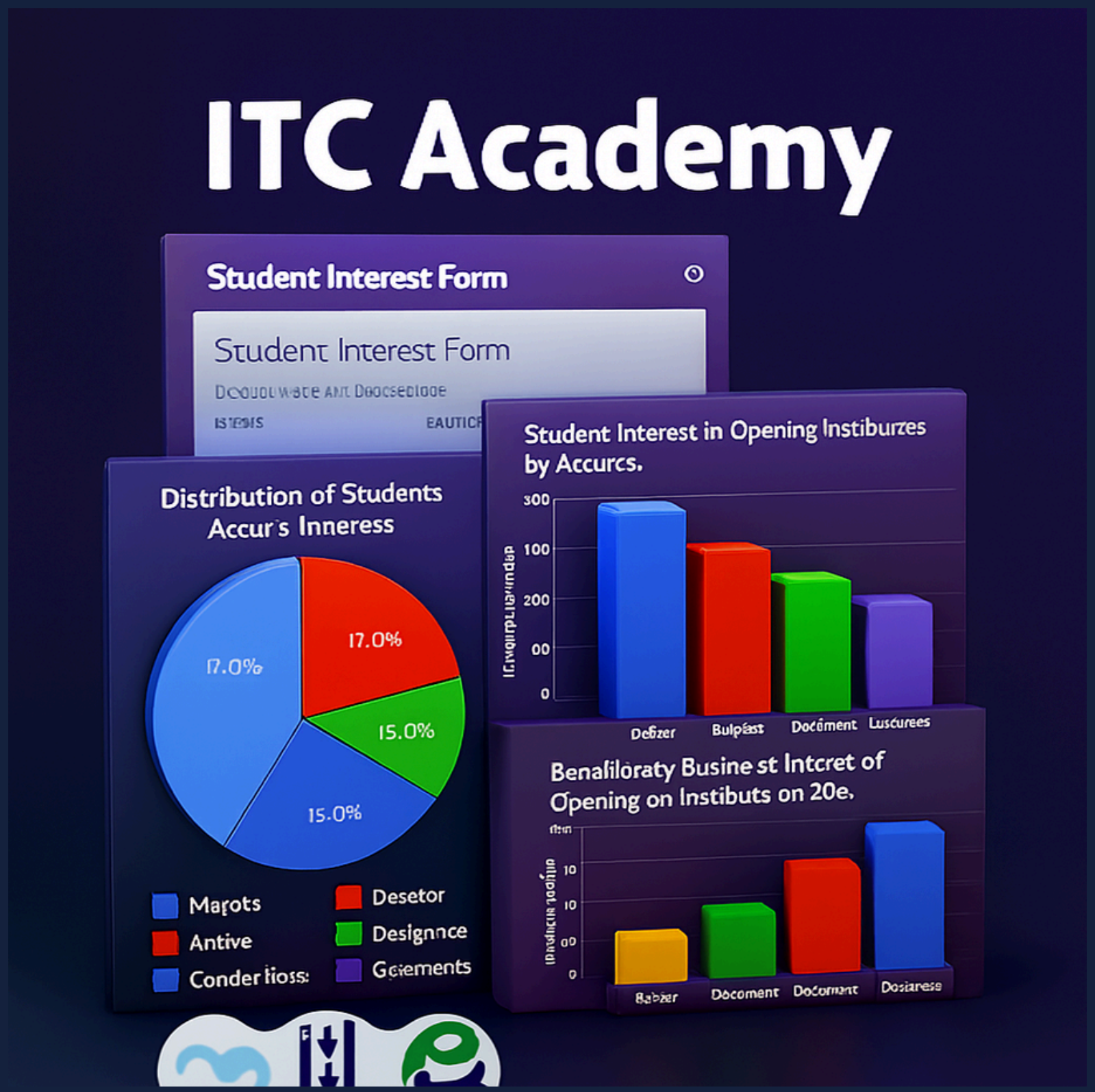


www.itcacademy.com

Table of Content:

1.Objective:

“Empowering graduates with the right skills, at the right place, for the right career.”

2.Data Description

Student_ID → Unique identifier for each respondent (anonymized).

District → The district where the graduate belongs (District A–E).

Course_Interest → The course chosen by the student (Web Development, Data Analyst, Software Developer, Testing Engineer, Digital Marketing).

Skill_Level → Current self-assessed skill level (Beginner, Intermediate, Advanced).

Availability → Student's availability for learning (Full-time, Part-time, Weekend).

Support_Expectation → Type of support expected (Placements, Certifications, Mentorship, Projects).

Year (optional if included) → Year of data collection (for trend analysis).

3.Basic Queries:

```python
# importing the python libraries for analyzing the dataset
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
# Load dataset
df = pd.read_csv("graduate_course_interest.csv")
```

```python
df
```

	Student_ID	District	Course_Interest	Skill_Level	Availability	Support_Expectation
0	1	District_E	Web Development	Beginner	Full-time	Placements
1	2	District_C	Digital Marketing	Intermediate	Full-time	Projects
2	3	District_C	Testing Engineer	Beginner	Part-time	Certifications
3	4	District_A	Testing Engineer	Beginner	Full-time	Placements
4	5	District_C	Digital Marketing	Beginner	Full-time	Mentorship
...	...	...	...	...	...	...
995	996	District_D	Digital Marketing	Advanced	Part-time	Placements
996	997	District_C	Digital Marketing	Intermediate	Full-time	Certifications
997	998	District_C	Software Developer	Advanced	Part-time	Mentorship
998	999	District_A	Data Analyst	Beginner	Full-time	Certifications
999	1000	District_C	Web Development	Beginner	Full-time	Certifications

1000 rows × 6 columns

```python
df.shape
```

```
(1000, 6)
```

Basic Queries:

```
df.columns
```

```
Index(['Student_ID', 'District', 'Course_Interest', 'Skill_Level',
       'Availability', 'Support_Expectation'],
      dtype='object')
```

```
df.isnull().sum()
```

```
Student_ID           0
District             0
Course_Interest      0
Skill_Level          0
Availability         0
Support_Expectation  0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Student_ID           1000 non-null   int64 
 1   District             1000 non-null   object
 2   Course_Interest      1000 non-null   object
 3   Skill_Level          1000 non-null   object
 4   Availability         1000 non-null   object
 5   Support_Expectation  1000 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.0+ KB
```

Basic Queries:

```python
# 1. Preview data
df.head(5)
```

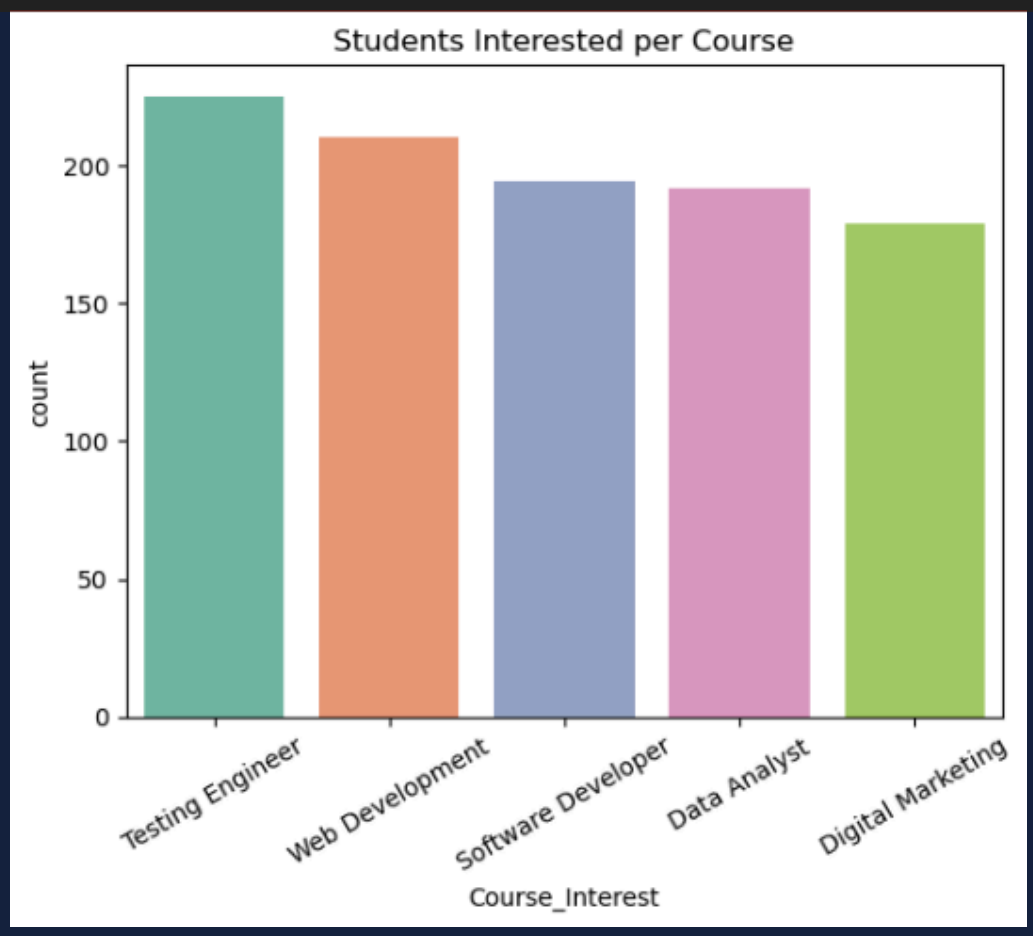
	Student_ID	District	Course_Interest	Skill_Level	Availability	Support_Expectation
0	1	District_E	Web Development	Beginner	Full-time	Placements
1	2	District_C	Digital Marketing	Intermediate	Full-time	Projects
2	3	District_C	Testing Engineer	Beginner	Part-time	Certifications
3	4	District_A	Testing Engineer	Beginner	Full-time	Placements
4	5	District_C	Digital Marketing	Beginner	Full-time	Mentorship

```python
# 2. Count students per course
course_counts = df['Course_Interest'].value_counts()
print(course_counts)
```

```
Course_Interest
Testing Engineer      225
Web Development       210
Software Developer    194
Data Analyst          192
Digital Marketing     179
Name: count, dtype: int64
```

4.Solutions via Visualization:

```
# Visualization
# as per the interest of students on several courses will help me to hire trainers according to the requirements

sns.countplot(data=df, x="Course_Interest", order=course_counts.index, palette="Set2")
plt.xticks(rotation=30)
plt.title("Students Interested per Course")
plt.show()
```



->Hire five full-time trainers for Testing Engineer

->Hire five full-time trainers for Web Development

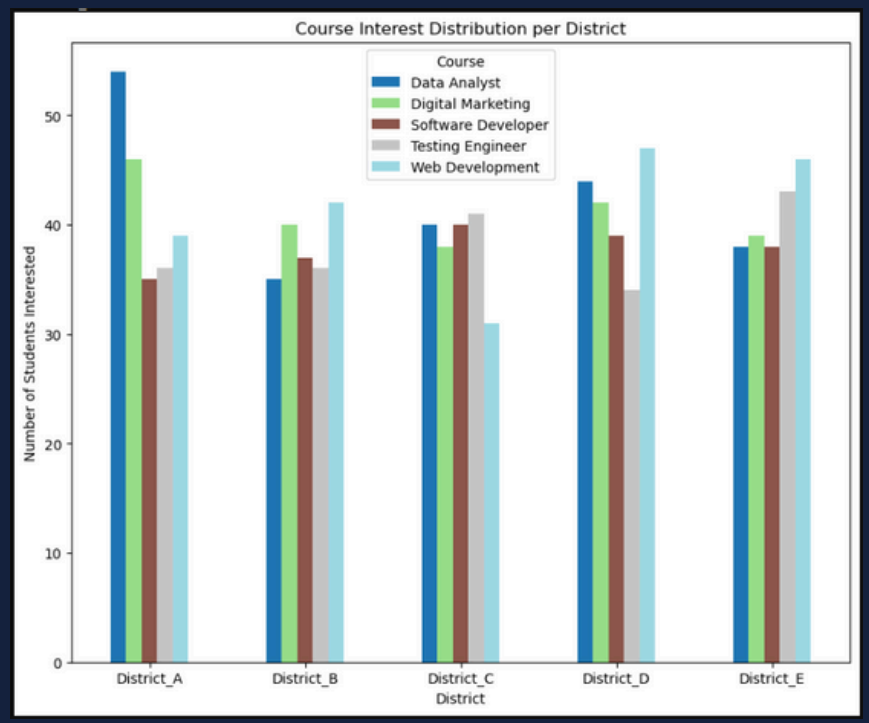
->Hire four full-time trainers for Software Developer

->Hire four full-time trainers for Data Analyst

->Hire four full-time trainers for Digital Marketing

District-Wise Course Demand

```python
# 1) District-Wise Course Demand

# Insight: Some districts have high demand for Data Analytics, others lean to Web Dev.
# Problem: Institutes give same courses everywhere → waste of resources.
# Solution: Allocate district-specific training programs.


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Sample data: 1000 students
np.random.seed(42)
districts = ['District_A', 'District_B', 'District_C', 'District_D', 'District_E']
courses = ['Web Development', 'Data Analyst', 'Software Developer', 'Testing Engineer', 'Digital Marketing']

# Create random data
data = {
    'Student_ID': range(1,1001),
    'District': np.random.choice(districts, 1000),
    'Course_Interest': np.random.choice(courses, 1000)
}

df = pd.DataFrame(data)

district_course = pd.crosstab(df['District'], df['Course_Interest'])
print(district_course)

district_course.plot(kind='bar', figsize=(10,8), colormap='tab20')
plt.title("Course Interest Distribution per District")
plt.xlabel("District")
plt.ylabel("Number of Students Interested")
plt.xticks(rotation=0)
plt.legend(title="Course")
plt.show()
```



->District_A has demand on Data Analyst

->District_B has demand on Web Developmentt

->District_C has demand on Testing Engineer

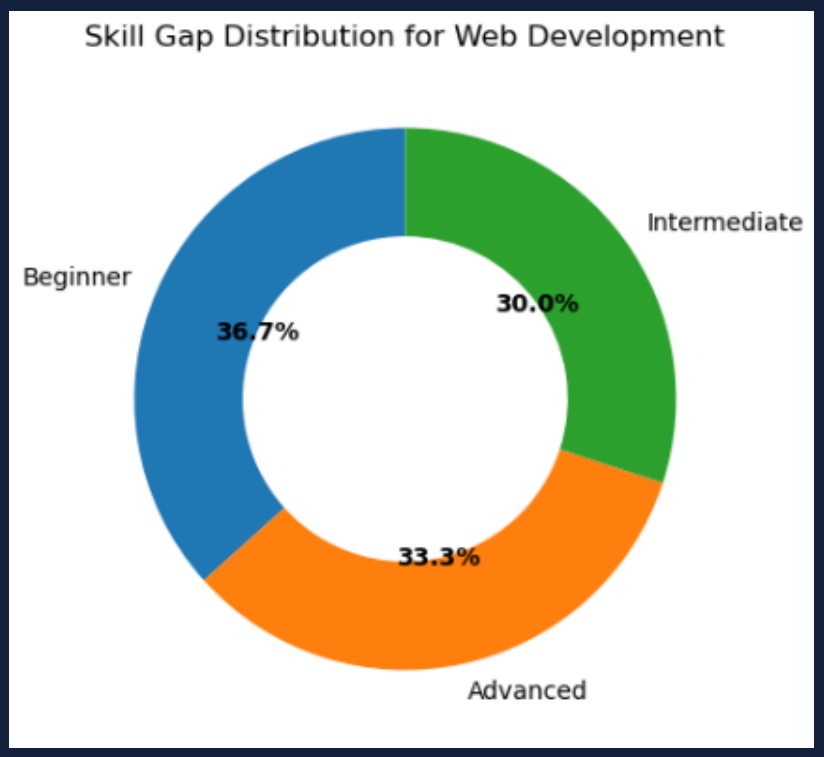
->District_D has demand on Web Development

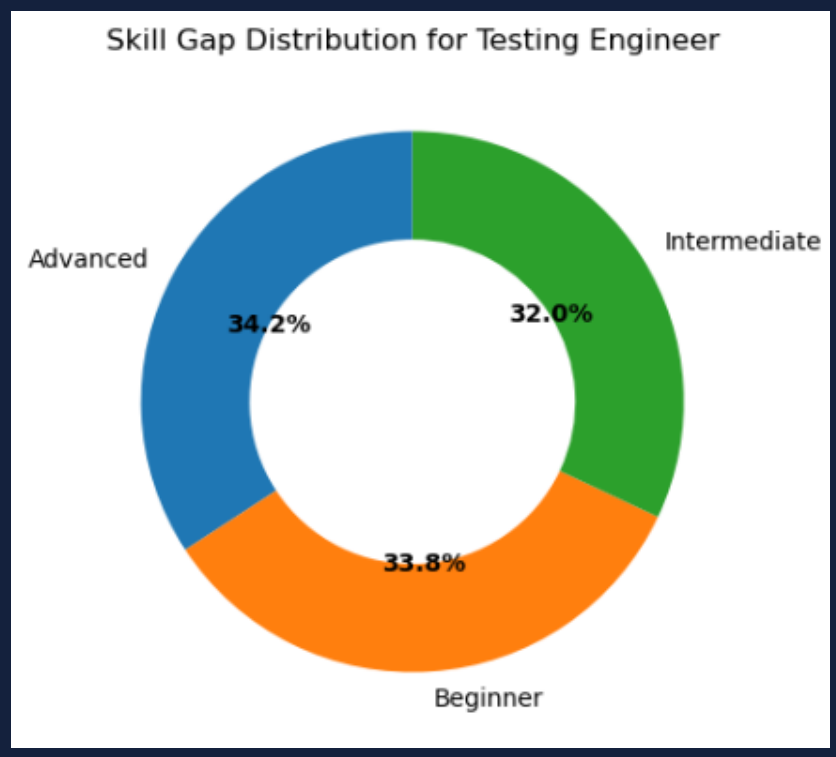
->District_E has demand on Web Development

-> Allocate district-wise specific Training Hubs

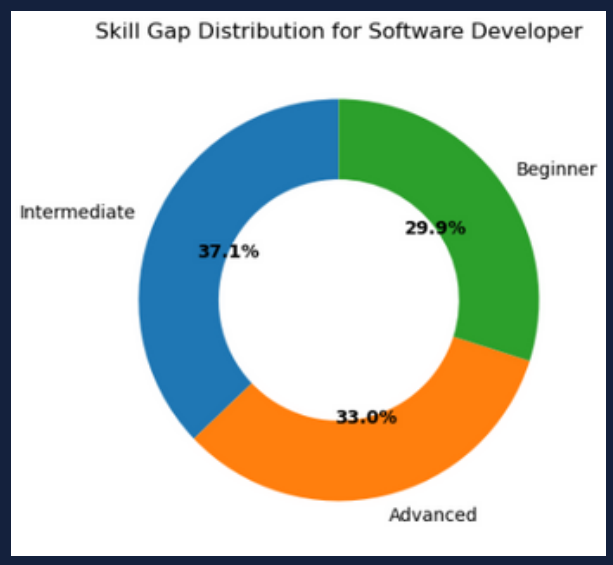
Skill-Gap Identification

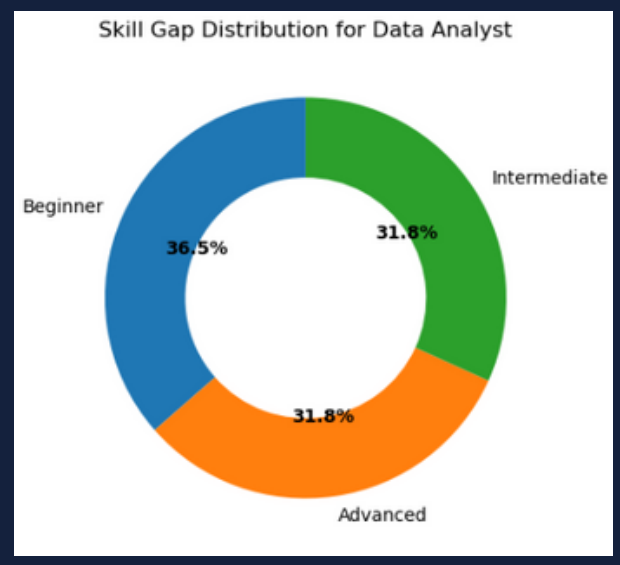
```python
# 2) Skill Gap Identification

# Insight: Many want Software Development but most are "Beginner."
# Problem: Leads to unemployment.
# Solution: Build pathway courses (Beginner → Advanced) + internships.

import matplotlib.pyplot as plt

courses = df["Course_Interest"].unique()

for course in courses:
    # Count skill levels for this course
    course_data = df[df["Course_Interest"] == course]["Skill_Level"].value_counts(normalize=True) * 100

    # Plot donut chart
    plt.figure(figsize=(5,5))
    wedges, texts, autotexts = plt.pie(
        course_data.values,
        labels=course_data.index,
        autopct="%1.1f%%",
        startangle=90,
        wedgeprops=dict(width=0.4)  # makes it donut
    )
    plt.setp(autotexts, size=10, weight="bold", color="black")
    plt.title(f"Skill Gap Distribution for {course}")
    plt.show()
```

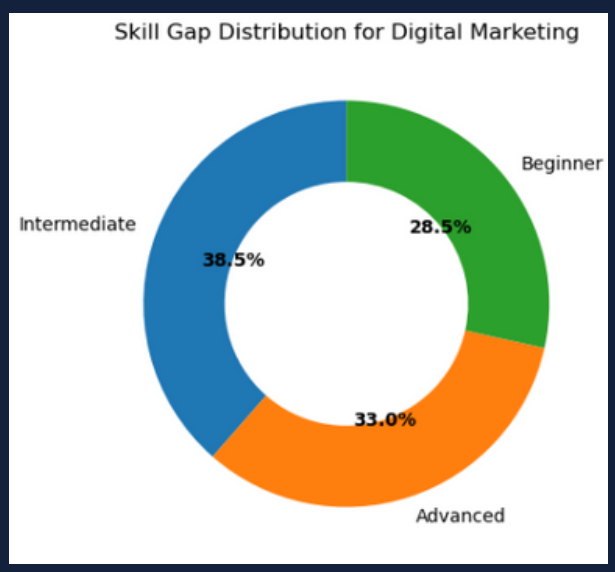




Skill-Gap Identification







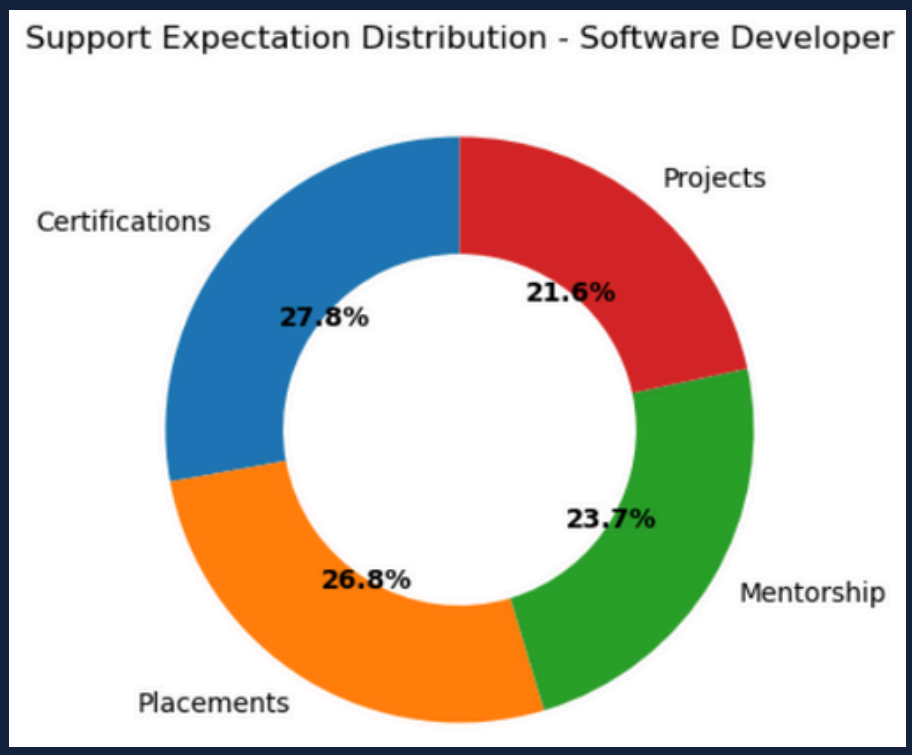
-> <u>Insight</u>: Many students want Software Development, but most are Beginners.

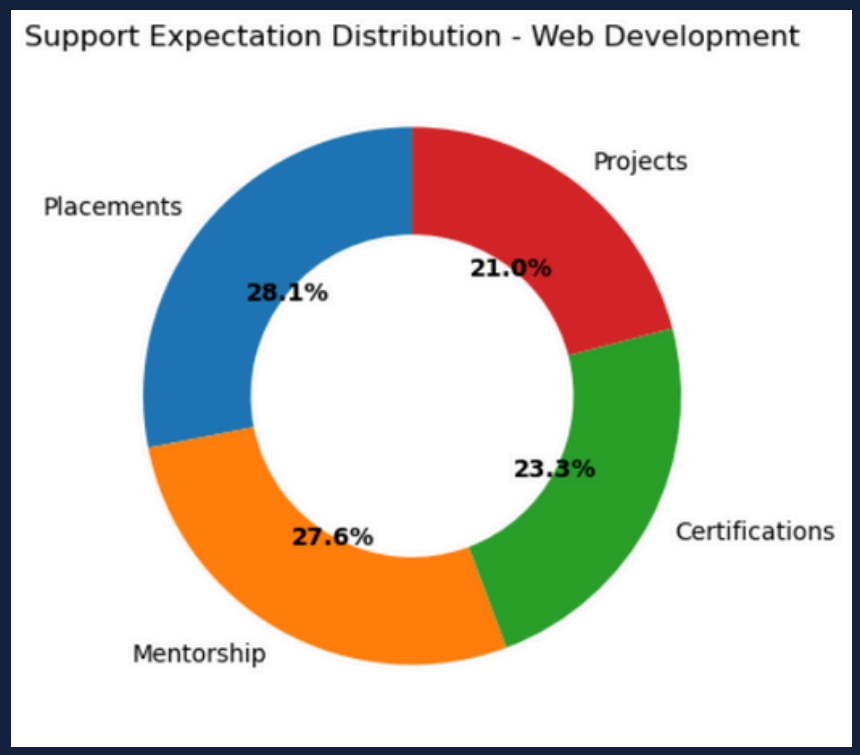
-> <u>Launch pathway programs</u>: Beginner → Intermediate → Advanced.

-> Add mini-projects + internships at each stage, so they become job-ready.

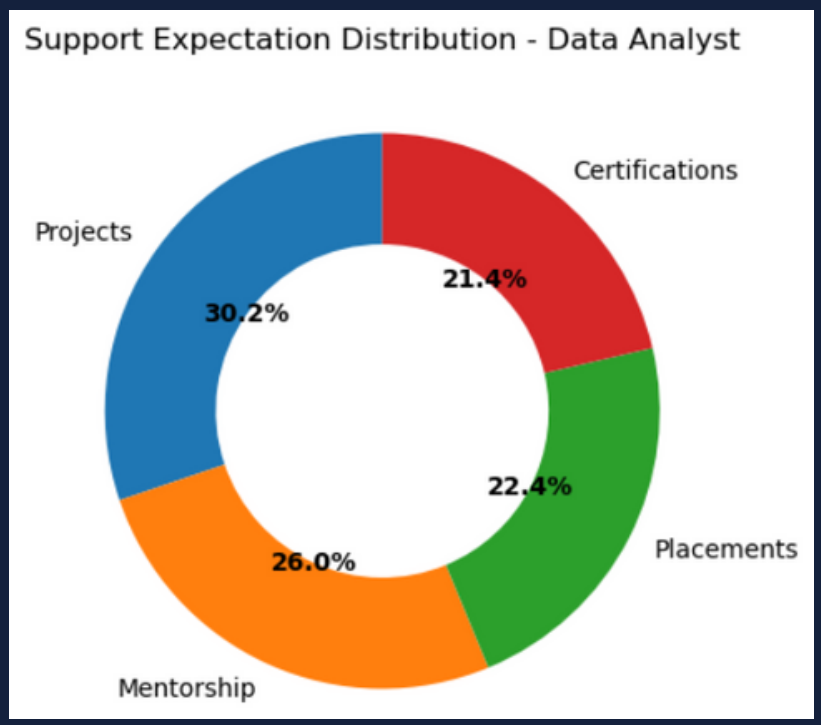
Support Expectation Mismatch

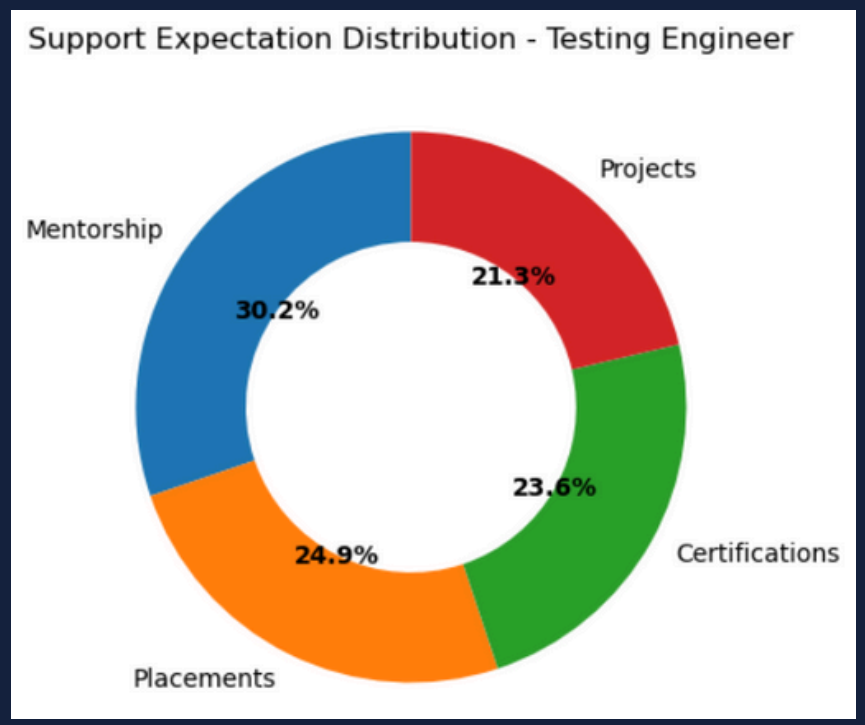
```python
import matplotlib.pyplot as plt

courses = df["Course_Interest"].unique()

for course in courses:
    # Filter data for this course
    course_data = df[df["Course_Interest"] == course]["Support_Expectation"].value_counts(normalize=True) * 100

    # Plot donut chart
    plt.figure(figsize=(5,5))
    wedges, texts, autotexts = plt.pie(
        course_data.values,
        labels=course_data.index,
        autopct="%1.1f%%",
        startangle=90,
        wedgeprops=dict(width=0.4)  # makes it donut
    )

    # Style the % text
    plt.setp(autotexts, size=10, weight="bold", color="black")

    plt.title(f"Support Expectation Distribution - {course}")
    plt.show()
```

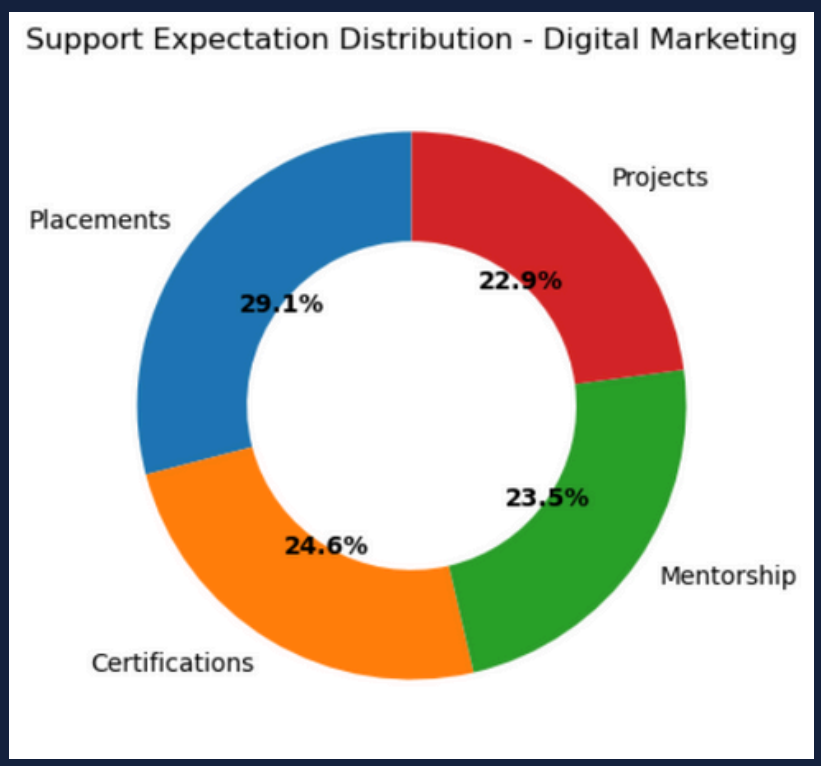




Support Expectation Mismatch







-> **Insight**: Students in some districts want placements, while others only care about certifications.

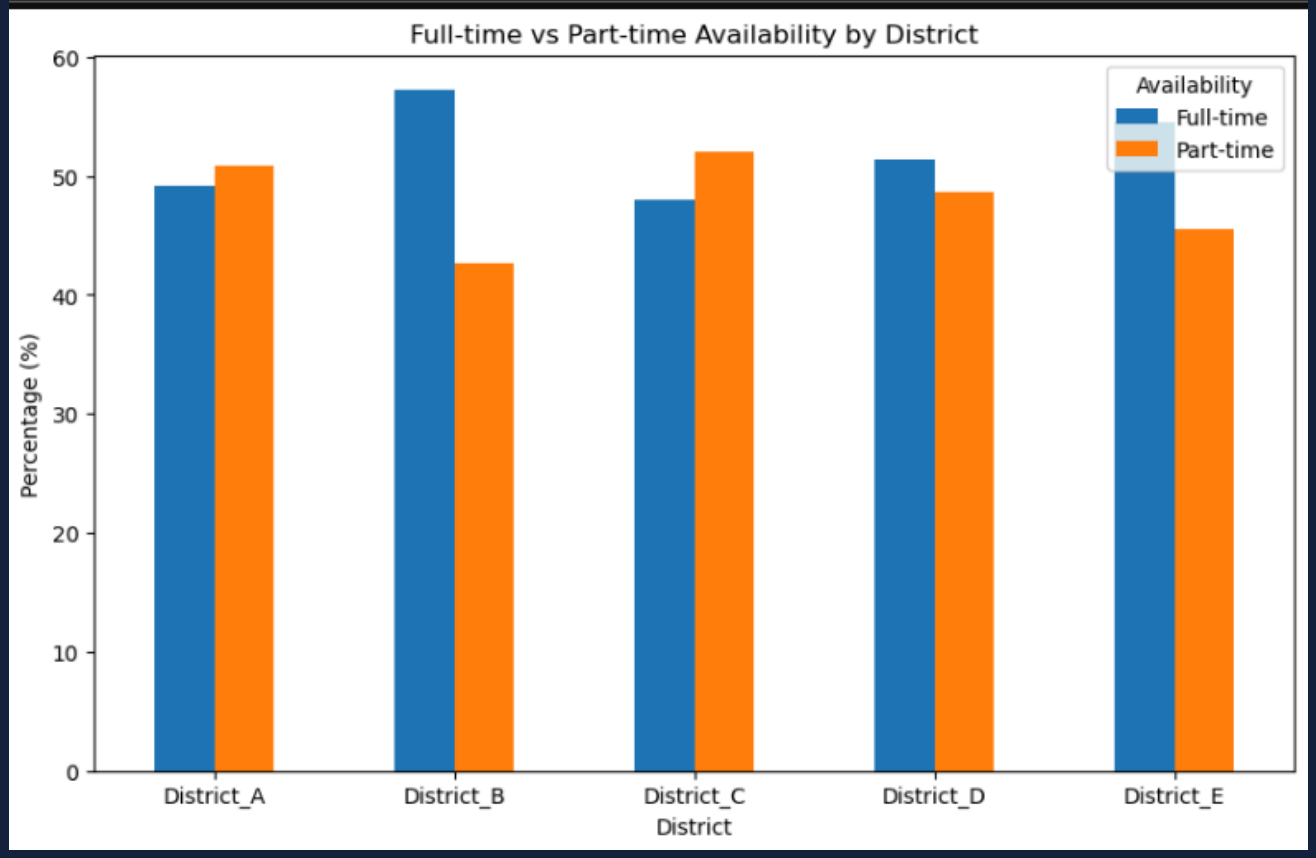
->Tie-up with local companies in placement-heavy districts

->Partner with online certification bodies in others.

->**Market differently**: “Guaranteed Placements” in District A, “Global Certifications” in District B.

Availability Patterns

```python
#6. Availability (Full-Time vs Part-Time, Per District)
availability_district = pd.crosstab(df["District"], df["Availability"], normalize="index") * 100

availability_district.plot(kind="bar", figsize=(10,6))
plt.title("Full-time vs Part-time Availability by District")
plt.ylabel("Percentage (%)")
plt.xticks(rotation=0)
plt.show()
```



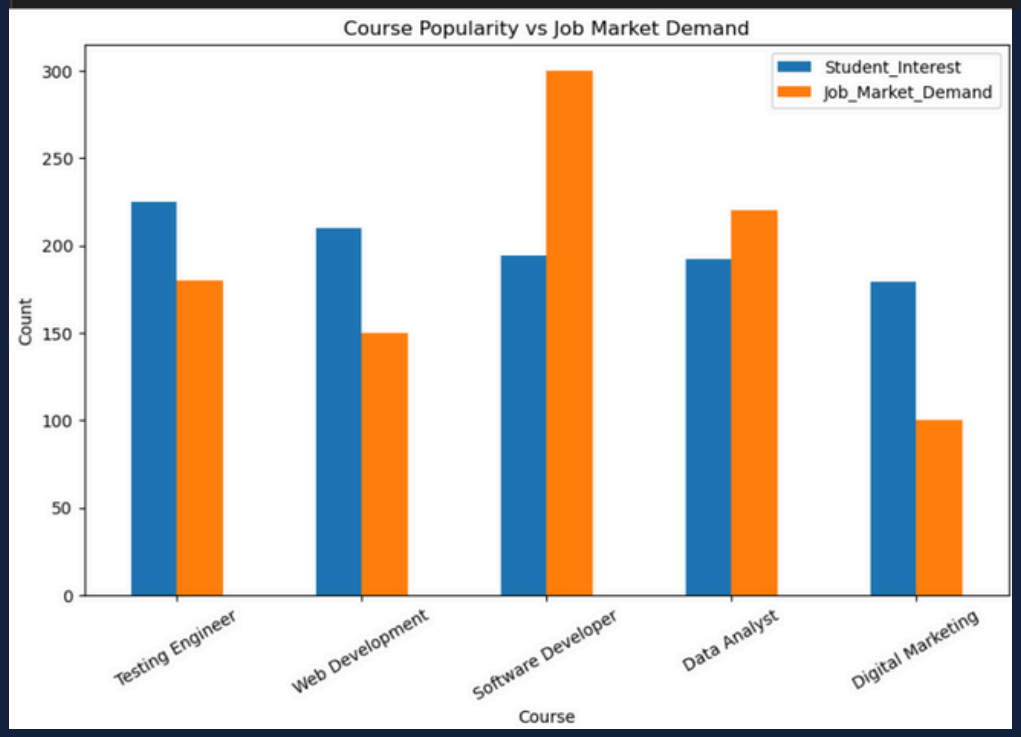
-> Insight: Some districts have more part-time learners (maybe already working), while others are full-time freshers.

-> Offer weekend / evening batches + self-paced online modules for part-timers.

-> Offer intensive bootcamps for freshers who are free full-time

Course vs Career Mapping

```python
# 5) Course-Career Mapping

# Insight: Some courses (e.g., Digital Marketing) are over-demanded but have low job openings.
# Problem: Students waste time → poor career outcomes.
# Solution: Provide market-aligned training guidance.
# Simulated job market demand (in reality, fetch from external sources)
job_demand = {
    "Web Development": 150,
    "Data Analyst": 220,
    "Software Developer": 300,
    "Testing Engineer": 180,
    "Digital Marketing": 100
}

student_interest = df["Course_Interest"].value_counts()

# Merge into DataFrame
compare_df = pd.DataFrame({
    "Course": student_interest.index,
    "Student_Interest": student_interest.values,
    "Job_Market_Demand": [job_demand[c] for c in student_interest.index]
})

compare_df.plot(x="Course", kind="bar", figsize=(10,6))
plt.title("Course Popularity vs Job Market Demand")
plt.ylabel("Count")
plt.xticks(rotation=30)
plt.show()
```



-> **Insight**: Some courses (like Digital Marketing) might be popular among students, but job openings are limited, compared to fields like Data Analytics.

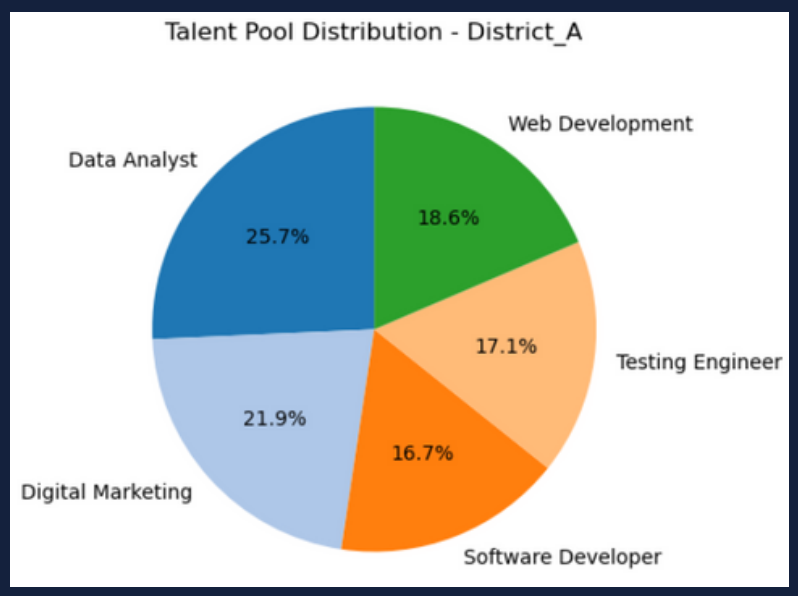
-> **Educate students** --> “Don’t just pick a popular course, pick a career-rich course.”

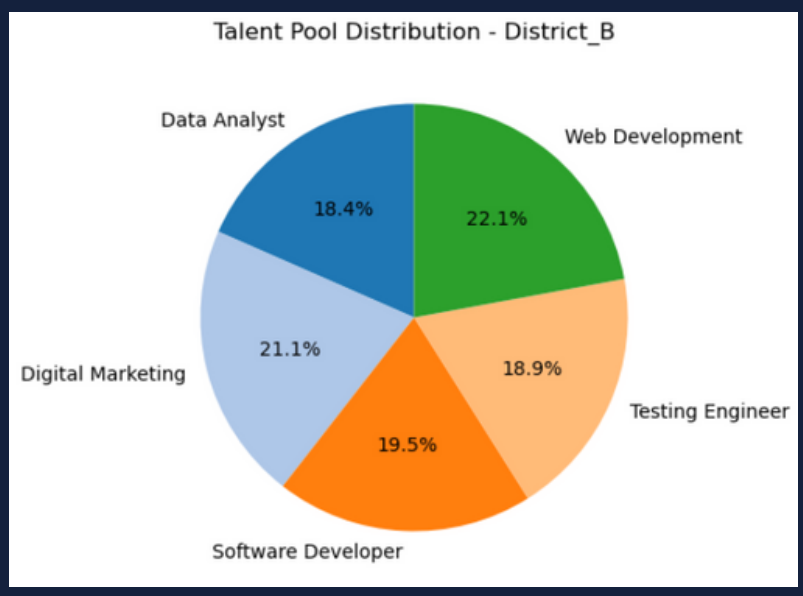
-> Provide career counselling sessions with market insights.

-> **Offer combo courses** (e.g., “Web Development + AI Basics” to make them stand out).

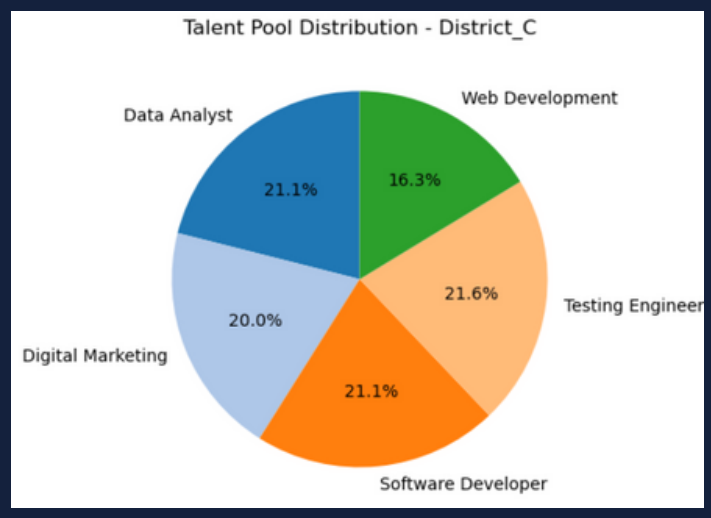
District-Wise Talent Pool

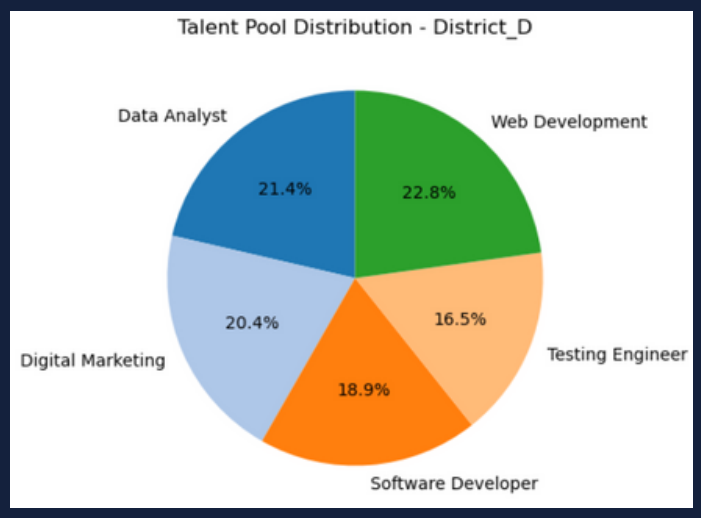
```python
# Count students per District × Course
talent_pool = (
    df.groupby(["District", "Course_Interest"])
    .size()
    .reset_index(name="Student_Count")
)

# Get unique districts
districts = talent_pool["District"].unique()

# Plot pie chart for each district
for district in districts:
    district_data = talent_pool[talent_pool["District"] == district]

    plt.figure(figsize=(5,5))
    plt.pie(
        district_data["Student_Count"],
        labels=district_data["Course_Interest"],
        autopct="%1.1f%%",
        startangle=90,
        colors=plt.cm.tab20.colors[:len(district_data)]
    )
    plt.title(f"Talent Pool Distribution - {district}")
    plt.show()
```

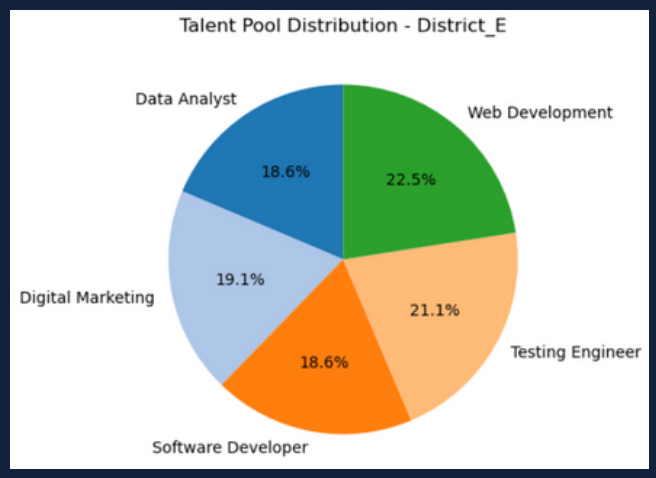




District-Wise Talent Pool







-> **Insight**: Companies don't know where to hire skilled students.

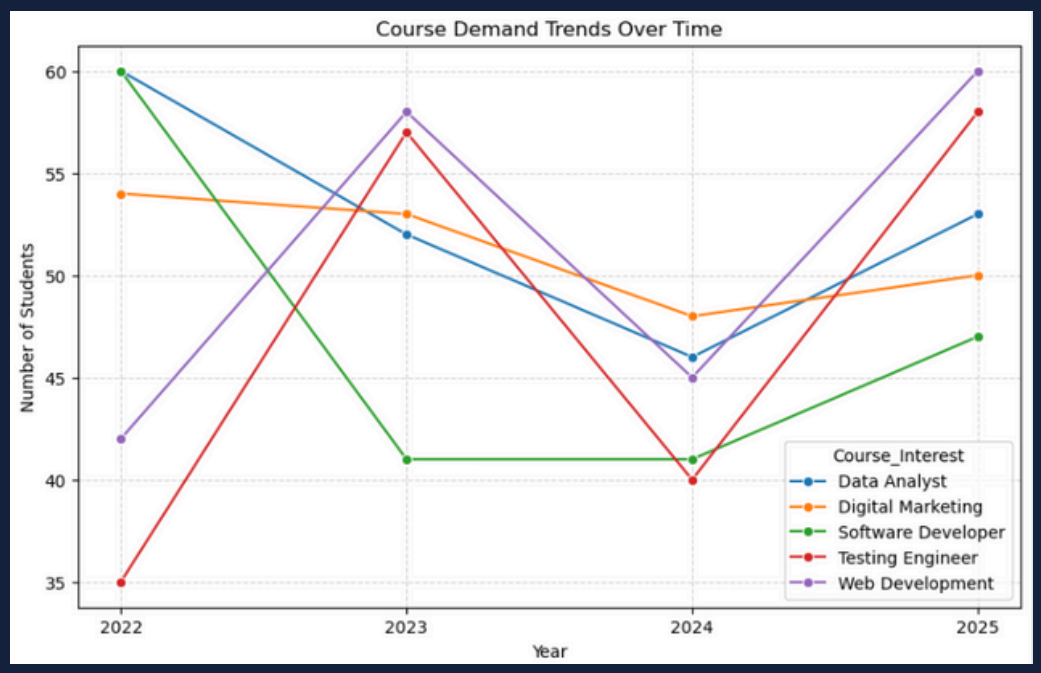
-> **Example**: "District_A has 200+ intermediate Data Analysts available."

-> Build District-wise Talent Maps.

-> Partner with IT parks / SMEs to place your students directly.

Trend Analysis Over Time

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Add fake year data (for demo)
df["Year"] = np.random.choice([2022, 2023, 2024, 2025], size=len(df))

# Group data
trend = df.groupby(["Year", "Course_Interest"]).size().reset_index(name="Count")

# Sort years properly
trend = trend.sort_values(by="Year")

# Plot line chart
plt.figure(figsize=(10,6))
sns.lineplot(data=trend, x="Year", y="Count", hue="Course_Interest", marker="o")

# Force x-axis to show all years
plt.xticks([2022, 2023, 2024, 2025])

plt.title("Course Demand Trends Over Time")
plt.ylabel("Number of Students")
plt.xlabel("Year")
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
```



-> **Insight**: Student interests change — today Web Dev, tomorrow AI/ML

-> Run yearly surveys.

-> Predict future demand → Launch AI/ML, Cybersecurity, Cloud courses before others do

-> Stay ahead of competitors by being trend-driven.

Business / Social Impact:

For Students -> Right courses, career-aligned, higher employability.

For Your Centre -> Save resources, design profitable district-specific strategies.

For Companies -> Easier access to trained graduates.

THANKYOU