SPRINT 4

TEAM ID	PNT2022TMID22144
PROJECT NAME	Skill Based Job Recommender Application

FINAL DELIVERY

CODE:

```
import streamlit as st
import streamlit.components.v1 as components
import pandas as pd
import numpy as np
import base64,random
import re,os
from ftfy import fix_text
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors
import time,datetime
from pyresparser import ResumeParser
from pdfminer3.layout import LAParams, LTTextBox
from pdfminer3.pdfpage import PDFPage
from pdfminer3.pdfinterp import PDFResourceManager
from pdfminer3.pdfinterp import PDFPageInterpreter
from pdfminer3.converter import TextConverter
import io,random
from streamlit_tags import st_tags
import streamlit.components.v1 as stc
from PIL import Image
import sqlite3
from Courses import
ds_course,web_course,android_course,ios_course,uiux_course,resume_videos,inter
view_videos
import plotly.express as px
from pathlib import Path
import hashlib

script_location = Path(__file__).absolute().parent
os.chdir(script_location)

def make_hashes(password):
    return hashlib.sha256(str.encode(password)).hexdigest()
```

```
def check_hashes(password,hashed_text):
    if make_hashes(password) == hashed_text:
        return hashed_text
    return False

conn = sqlite3.connect('data.db')
c = conn.cursor()
def create_usertable():
    c.execute('CREATE TABLE IF NOT EXISTS userstable(username TEXT,password
TEXT)')


def add_userdata(username,password):
    c.execute('INSERT INTO userstable(username,password) VALUES
(?,?)',(username,password))
    conn.commit()

def login_user(username,password):
    c.execute('SELECT * FROM userstable WHERE username =? AND password =
?',(username,password))
    data = c.fetchall()
    return data


def pdf_reader(file):
    resource_manager = PDFResourceManager()
    fake_file_handle = io.StringIO()
    converter = TextConverter(resource_manager, fake_file_handle,
laparams=LAParams())

    page_interpreter = PDFPageInterpreter(resource_manager, converter)
    with open(file, 'rb') as fh:
        for page in PDFPage.get_pages(fh,
                                      caching=True,
                                      check_extractable=True):
            page_interpreter.process_page(page)
            print(page)
        text = fake_file_handle.getvalue()

    # close open handles
    converter.close()
    fake_file_handle.close()
    return text

def show_pdf(file_path):
    with open(file_path, "rb") as f:
        base64_pdf = base64.b64encode(f.read()).decode('utf-8')
    # pdf_display = f'<embed src="data:application/pdf;base64,{base64_pdf}"
width="700" height="1000" type="application/pdf">'
```

```
    pdf_display = F'<iframe src="data:application/pdf;base64,{base64_pdf}"
width="700" height="1000" type="application/pdf"></iframe>'
    st.markdown(pdf_display, unsafe_allow_html=True)

def course_recommender(course_list):
    st.subheader("**Courses & Certificates🎓 Recommendations**")
    c = 0
    rec_course = []
    no_of_reco = st.slider('Choose Number of Course Recommendations:', 1, 10,
4)
    random.shuffle(course_list)
    for c_name, c_link in course_list:
        c += 1
        st.markdown(f"({c}) [{c_name}]({c_link})")
        rec_course.append(c_name)
        if c == no_of_reco:
            break
    return rec_course


def searchbox():
    prefinal = []
    skill_ip = []
    skill_ip = str(st.text_area('Enter the Skills and press cntrl + enter'))
    butt = st.button("search")
    if butt:
        prefinal = skill_ip.split('\n')
        recommender(prefinal)

def rsl(rsd):
    resskill = rsd['skills']
    recommender(resskill)

def convertTuple(tup):
        # initialize an empty string
    str = ''
    for item in tup:
        str = str + item
    return str

def recommender(rskl):
    jskills = []
    jskills.append(' '.join(word for word in rskl))
    org_name_clean = jskills

    stopw  = set(stopwords.words('english'))
    df1 =pd.read_csv('job_final.csv')
    df1['test']=df1['Job_Description'].apply(lambda x: ' '.join([word for word
in str(x).split() if len(word)>2 and word not in (stopw)]))
```

```
    def ngrams(string, n=3):
        string = fix_text(string) # fix text
        string = string.encode("ascii", errors="ignore").decode() #remove non
ascii chars
        string = string.lower()
        chars_to_remove = [")","(",".","|","[","]","{","}","'"]
        rx = '[' + re.escape(''.join(chars_to_remove)) + ']'
        string = re.sub(rx, '', string)
        string = string.replace('&', 'and')
        string = string.replace(',', ' ')
        string = string.replace('-', ' ')
        string = string.title() # normalise case - capital at start of each
word
        string = re.sub(' +',' ',string).strip() # get rid of multiple spaces
and replace with a single
        string = ' '+ string +' ' # pad names for ngrams...
        string = re.sub(r'[,-./]|\sBD',r'', string)
        ngrams = zip(*[string[i:] for i in range(n)])
        return [''.join(ngram) for ngram in ngrams]

    vectorizer = TfidfVectorizer(min_df=1, analyzer=ngrams, lowercase=False)
    tfidf = vectorizer.fit_transform(org_name_clean)


    def getNearestN(query):
        queryTFIDF_ = vectorizer.transform(query)
        distances, indices = nbrs.kneighbors(queryTFIDF_)
        return distances, indices

    nbrs = NearestNeighbors(n_neighbors=1, n_jobs=-1).fit(tfidf)
    unique_org = (df1['test'].values)
    distances, indices = getNearestN(unique_org)
    unique_org = list(unique_org)
    matches = []
    for i,j in enumerate(indices):
        dist=round(distances[i][0],2)

        temp = [dist]
        matches.append(temp)
    matches = pd.DataFrame(matches, columns=['Match confidence'])
    df1['match']=matches['Match confidence']
    df11=df1.sort_values('match')
    df2=df11[['Position', 'Company','Location','url']].head(10).reset_index()
    # st.table(df2)

    list_template = """
    <div style='color:#fff'>
```

```
    <h2>{}</h2>
    <h3>{}</h3>
    <h4>{}</h4>
    </div>
    """
    for i in range(len(df2)):
        jt = df2.loc[i, "Position"]
        cp = df2.loc[i, "Company"]
        lc = df2.loc[i, "Location"]
        jurl = df2.loc[i,"url"]
        st.markdown(list_template.format(jt,cp,lc),unsafe_allow_html=True)
        with st.expander("Apply for job"):
            stc.html(jurl)


st.set_page_config(
   page_title="Job Recommender",
)
def run():
    st.title("Job Recommender")
# st.markdown('''<h4 style='text-align: left; color: #d73b5c;'>* Upload your
resume, and get smart recommendation based on it."</h4>''',
#             unsafe_allow_html=True)

    st.sidebar.markdown("# Job Recommender")
    activities = ["Sign Up","Log In","Search For Job Recommendation","Upload
Resume"]
    choice = st.sidebar.selectbox("Choose: ", activities)
    if choice == "Sign Up":
        st.subheader("Create an Account")
        new_user = st.text_input('Username')
        new_passwd = st.text_input('Password',type='password')
        components.html('''
                <style>
                #place{
                    posistion:fixed;
                    bottom:0;
                }
                </style>
                <div id="place">
                    <script>
                        window.watsonAssistantChatOptions = {integrationID:
"0bb96b92-4e98-44c7-9dab-3a5fe2ff8562",region: "au-syd", serviceInstanceID:
"e5babddc-2ad5-4eac-a0c5-6dad126622cb", onLoad: function(instance) {
instance.render(); }};
                        setTimeout(function(){const
t=document.createElement('script'); t.src="https://web-
chat.global.assistant.watson.appdomain.cloud/versions/" +
```

```
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js"; document.head.appendChild(t);});
                    </script
                </div>
                ''',height=600,)
        if st.button('SignUp'):
            create_usertable()
            add_userdata(new_user,make_hashes(new_passwd))
            st.success("You have successfully created an account.Go to the
Login Menu to login")

    elif choice == "Log In":
        user = st.sidebar.text_input('Username')
        passwd = st.sidebar.text_input('Password',type='password')
        if st.sidebar.checkbox('Login') :
            create_usertable()
            hashed_pswd = make_hashes(passwd)
            result = login_user(user,check_hashes(passwd,hashed_pswd))
            if result:
                st.success("Logged In as {}".format(user))

                    # Tasks For Only Logged In Users

    elif choice == 'Search For Job Recommendation':
        searchbox()

    else:
        pdf_file = st.file_uploader("Choose your Resume", type=["pdf"])

        if pdf_file is not None:
            # with st.spinner('Uploading your Resume....'):
            #     time.sleep(4)
            # script_location = Path(__file__).absolute().parent
            # os.chdir(script_location)
            save_image_path = './Uploaded_Resumes' +pdf_file.name
            with open(save_image_path, "wb") as f:
                f.write(pdf_file.getbuffer())
            show_pdf(save_image_path)
            resume_data = ResumeParser(save_image_path).get_extracted_data()
            if resume_data:
                ## Get the whole resume data
                resume_text = pdf_reader(save_image_path)
                st.header("**Resume Analysis**")
                st.success("Hello "+ resume_data['name'])
                st.subheader("**Your Basic info**")

                try:
                    st.text('Name: '+resume_data['name'])
```

```
                    st.text('Email: ' + resume_data['email'])
                    st.text('Contact: ' + resume_data['mobile_number'])
                    st.text('Resume pages: '+str(resume_data['no_of_pages']))
                except:
                    pass
                cand_level = ''
                if resume_data['no_of_pages'] == 1:
                    cand_level = "Fresher"
                    st.markdown( '''<h4 style='text-align: left; color:
#d73b5c;'>You are looking Fresher.</h4>''',unsafe_allow_html=True)
                elif resume_data['no_of_pages'] == 2:
                    cand_level = "Intermediate"
                    st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>You are at intermediate level!</h4>''',unsafe_allow_html=True)
                elif resume_data['no_of_pages'] >=3:
                    cand_level = "Experienced"
                    st.markdown('''<h4 style='text-align: left; color:
#fba171;'>You are at experience level!''',unsafe_allow_html=True)

                st.subheader("**Skills Recommendation💡**")
                ## Skill shows
                keywords = st_tags(label='### Skills that you have',
                text='See our skills recommendation',
                    value=resume_data['skills'],key = '1')

                ##  recommendation
                ds_keyword = ['tensorflow','keras','pytorch','machine
learning','deep Learning','flask','streamlit']
                web_keyword = ['react', 'django', 'node jS', 'react js',
'php', 'laravel', 'magento', 'wordpress',
                                'javascript', 'angular js', 'c#', 'flask']
                android_keyword = ['android','android
development','flutter','kotlin','xml','kivy']
                ios_keyword = ['ios','ios development','swift','cocoa','cocoa
touch','xcode']
                uiux_keyword = ['ux','adobe
xd','figma','zeplin','balsamiq','ui','prototyping','wireframes','storyframes',
'adobe photoshop','photoshop','editing','adobe
illustrator','illustrator','adobe after effects','after effects','adobe
premier pro','premier pro','adobe
indesign','indesign','wireframe','solid','grasp','user research','user
experience']

                recommended_skills = []
                reco_field = ''
                rec_course = ''
                ## Courses recommendation
                for i in resume_data['skills']:
```

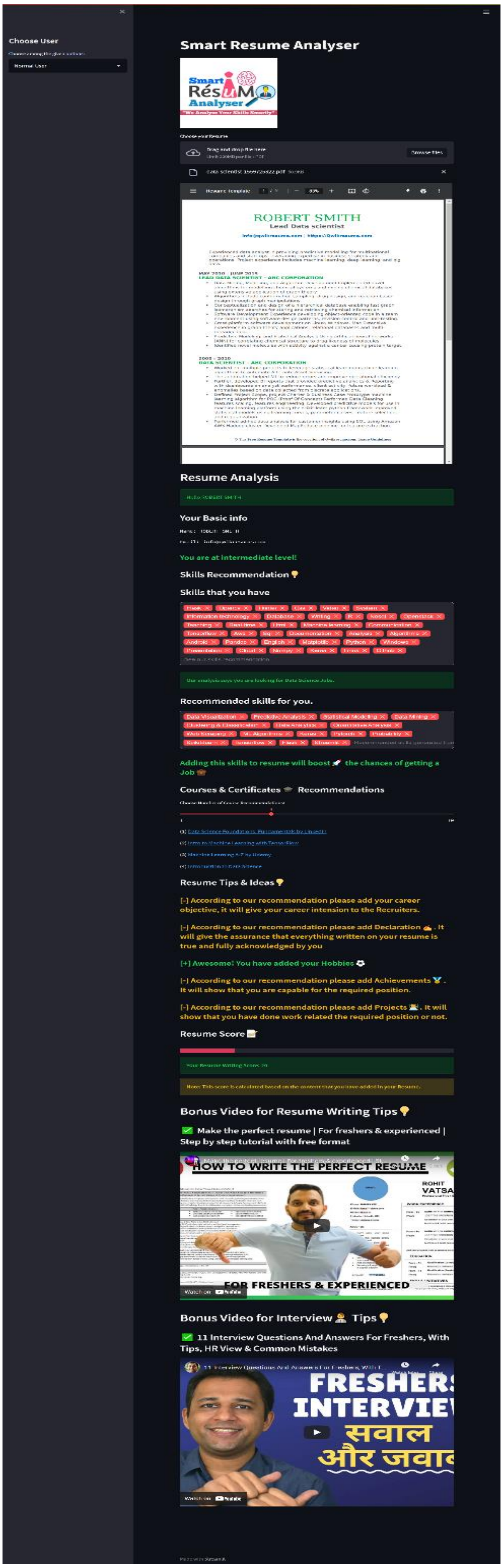
```
                    ## Data science recommendation
                    if i.lower() in ds_keyword:
                        print(i.lower())
                        reco_field = 'Data Science'
                        st.success("** Our analysis says you are looking for
Data Science Jobs.**")
                        recommended_skills = ['Data Visualization','Predictive
Analysis','Statistical Modeling','Data Mining','Clustering &
Classification','Data Analytics','Quantitative Analysis','Web Scraping','ML
Algorithms','Keras','Pytorch','Probability','Scikit-
learn','Tensorflow',"Flask",'Streamlit']
                        recommended_keywords = st_tags(label='### Recommended
skills for you.',
                        text='Recommended skills generated from
System',value=recommended_skills,key = '2')
                        st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>Adding this skills to resume will boost🚀 the chances of getting a
Job💼</h4>''',unsafe_allow_html=True)
                        rec_course = course_recommender(ds_course)
                        break

                    ## Web development recommendation
                    elif i.lower() in web_keyword:
                        print(i.lower())
                        reco_field = 'Web Development'
                        st.success("** Our analysis says you are looking for
Web Development Jobs **")
                        recommended_skills = ['React','Django','Node
JS','React JS','php','laravel','Magento','wordpress','Javascript','Angular
JS','c#','Flask','SDK']
                        recommended_keywords = st_tags(label='### Recommended
skills for you.',
                        text='Recommended skills generated from
System',value=recommended_skills,key = '3')
                        st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>Adding this skills to resume will boost🚀 the chances of getting a
Job💼</h4>''',unsafe_allow_html=True)
                        rec_course = course_recommender(web_course)
                        break

                    ## Android App Development
                    elif i.lower() in android_keyword:
                        print(i.lower())
                        reco_field = 'Android Development'
                        st.success("** Our analysis says you are looking for
Android App Development Jobs **")
                        recommended_skills = ['Android','Android
development','Flutter','Kotlin','XML','Java','Kivy','GIT','SDK','SQLite']
```

```
                        recommended_keywords = st_tags(label='### Recommended
skills for you.',
                        text='Recommended skills generated from
System',value=recommended_skills,key = '4')
                        st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>Adding this skills to resume will boost🚀 the chances of getting a
Job💼</h4>''',unsafe_allow_html=True)
                        rec_course = course_recommender(android_course)
                        break

                    ## IOS App Development
                    elif i.lower() in ios_keyword:
                        print(i.lower())
                        reco_field = 'IOS Development'
                        st.success("** Our analysis says you are looking for
IOS App Development Jobs **")
                        recommended_skills = ['IOS','IOS
Development','Swift','Cocoa','Cocoa Touch','Xcode','Objective-
C','SQLite','Plist','StoreKit',"UI-Kit",'AV Foundation','Auto-Layout']
                        recommended_keywords = st_tags(label='### Recommended
skills for you.',
                        text='Recommended skills generated from
System',value=recommended_skills,key = '5')
                        st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>Adding this skills to resume will boost🚀 the chances of getting a
Job💼</h4>''',unsafe_allow_html=True)
                        rec_course = course_recommender(ios_course)
                        break

                    ## Ui-UX Recommendation
                    elif i.lower() in uiux_keyword:
                        print(i.lower())
                        reco_field = 'UI-UX Development'
                        st.success("** Our analysis says you are looking for
UI-UX Development Jobs **")
                        recommended_skills = ['UI','User Experience','Adobe
XD','Figma','Zeplin','Balsamiq','Prototyping','Wireframes','Storyframes','Adob
e Photoshop','Editing','Illustrator','After Effects','Premier
Pro','Indesign','Wireframe','Solid','Grasp','User Research']
                        recommended_keywords = st_tags(label='### Recommended
skills for you.',
                        text='Recommended skills generated from
System',value=recommended_skills,key = '6')
                        st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>Adding this skills to resume will boost🚀 the chances of getting a
Job💼</h4>''',unsafe_allow_html=True)
                        rec_course = course_recommender(uiux_course)
                        break
```

```
                ### Resume writing recommendation
                st.subheader("**Resume Tips & Ideas💡**")
                resume_score = 0
                if 'Objective' in resume_text:
                    resume_score = resume_score+20
                    st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>[+] Awesome! You have added
Objective</h4>''',unsafe_allow_html=True)
                else:
                    st.markdown('''<h4 style='text-align: left; color:
#fabc10;'>[-] According to our recommendation please add your career
objective, it will give your career intension to the
Recruiters.</h4>''',unsafe_allow_html=True)

                if 'Declaration'  in resume_text:
                    resume_score = resume_score + 20
                    st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>[+] Awesome! You have added
Delcaration✍/h4>''',unsafe_allow_html=True)
                else:
                    st.markdown('''<h4 style='text-align: left; color:
#fabc10;'>[-] According to our recommendation please add Declaration✍. It
will give the assurance that everything written on your resume is true and
fully acknowledged by you</h4>''',unsafe_allow_html=True)

                if 'Hobbies' or 'Interests'in resume_text:
                    resume_score = resume_score + 20
                    st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>[+] Awesome! You have added your
Hobbies⚽</h4>''',unsafe_allow_html=True)
                else:
                    st.markdown('''<h4 style='text-align: left; color:
#fabc10;'>[-] According to our recommendation please add Hobbies⚽. It will
show your persnality to the Recruiters and give the assurance that you are fit
for this role or not.</h4>''',unsafe_allow_html=True)

                if 'Achievements' in resume_text:
                    resume_score = resume_score + 20
                    st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>[+] Awesome! You have added your Achievements🏅
</h4>''',unsafe_allow_html=True)
                else:
                    st.markdown('''<h4 style='text-align: left; color:
#fabc10;'>[-] According to our recommendation please add Achievements🏅. It
```

```
will show that you are capable for the required
position.</h4>''',unsafe_allow_html=True)

                if 'Projects' in resume_text:
                    resume_score = resume_score + 20
                    st.markdown('''<h4 style='text-align: left; color:
#1ed760;'>[+] Awesome! You have added your Projects👦💻
</h4>''',unsafe_allow_html=True)
                else:
                    st.markdown('''<h4 style='text-align: left; color:
#fabc10;'>[-] According to our recommendation please add Projects👦💻. It
will show that you have done work related the required position or
not.</h4>''',unsafe_allow_html=True)

                st.subheader("**Resume Score📝**")
                st.markdown(
                    """
                    <style>
                        .stProgress > div > div > div > div {
                            background-color: #d73b5c;
                        }
                    </style>""",
                    unsafe_allow_html=True,
                )
                my_bar = st.progress(0)
                score = 0
                for percent_complete in range(resume_score):
                    score +=1
                    time.sleep(0.1)
                    my_bar.progress(percent_complete + 1)
                st.success('** Your Resume Writing Score: ' + str(score)+'**')
                st.warning("** Note: This score is calculated based on the
content that you have added in your Resume. **")
                st.balloons()
                rsd = resume_data
                rsl(rsd)



            else:
                st.error('Something went wrong..')


run()
```

OUTPUT:


Choose User
Normal User
Smart Resume Analyser
Smart Résumé Analyser
ROBERT SMITH
Lead Data scientist
Resume Analysis
Your Basic info
You are at intermediate level!
Skills Recommendation
Skills that you have
Recommended skills for you.
Adding this skills to resume will boost the chances of getting a Job
Courses & Certificates Recommendations
Resume Tips & Ideas
[-] According to our recommendation please add your career objective, it will give your career intension to the Recruiters.
[-] According to our recommendation please add Declaration. It will give the assurance that everything written on your resume is true and fully acknowledged by you
[+] Awesome! You have added your Hobbies
[-] According to our recommendation please add Achievements. It will show that you are capable for the required position.
[-] According to our recommendation please add Projects. It will show that you have done work related the required position or not.
Resume Score
Bonus Video for Resume Writing Tips
Make the perfect resume | For freshers & experienced | Step by step tutorial with free format
HOW TO WRITE THE PERFECT RESUME
FOR FRESHERS & EXPERIENCED
Bonus Video for Interview Tips
11 Interview Questions And Answers For Freshers, With Tips, HR View & Common Mistakes
सवाल

Choose User

Choose among the given options:

Admin

Smart Resume Analyser

Welcome to Admin Side

Username

machine_learning_hub

Password

Login

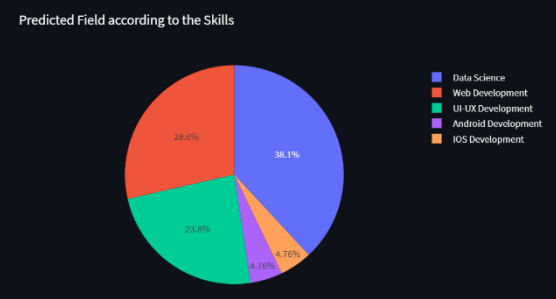
Welcome Kushal

User's Data

	ID	Name	Email	Resume Score	Timestamp
32	33	Javascript Developer	info@qwikresume.com	40	2022-03-05_11:4
33	34	ROBERT SMITH	info@qwikresume.com	20	2022-03-05_12:1
34	35	ROBERT SMITH	info@qwikresumc.com	20	2022-03-05_12:2
35	36	Android Developer	info@qwikresume.com	40	2022-03-05_10:2
36	37	art director	hello@allisonbeer.com	20	2022-03-05_10:5
37	38	art director	hello@allisonbeer.com	20	2022-03-05_10:5
38	39	art director	hello@allisonbeer.com	20	2022-03-05_11:0
39	40	ROBERT SMITH	info@qwikresume.com	20	2022-03-06_03:1
40	41	Kushal Bhavsar	kushalbhavsar58@gmail....	80	2022-03-06_03:1
41	42	ROBERT SMITH	info@qwikresume.com	20	2022-03-06_03:1

Download Report

Pie-Chart for Predicted Field Recommendations



Pie-Chart for User's Experienced Level