Big Data-Driven Insights: Trends in IT Skills for Education

Leveraging Job Market Data to Guide Educational Content and Marketing Strategies

Abdurehman Asfaw (448665)
Bhupender Bhupender (466758)
Zaid Alotel (468411)

Introduction

Project Overview :

- The goal was to help an educational company identify key IT skills in high demand and understand their value for marketing and curriculum design.
- A Proof-of-Concept (PoC) project designed to analyze IT job offers data from JustJoinIT, Poland's leading IT job board.
- The analysis explored trends across programming languages, technologies, and job categories over two years of data.
- Delivered a dynamic dashboard enabling the client to perform ongoing trend analysis.

Key Tools and Technologies:

- Azure
- GCP
- Spark
- Dask
- Kaggle

Data Preprocessing:

- Tools Used: Azure, Databricks, GCP, Spark, Dask and Parquet.

- Steps:
  - Transformed data into structured formats using Spark for scalability and speed.
  - Converted processed data into Parquet format for efficient storage and querying.
  - Analyzed IT skill demand across job roles, technologies, and locations.

Dashboard Creation:

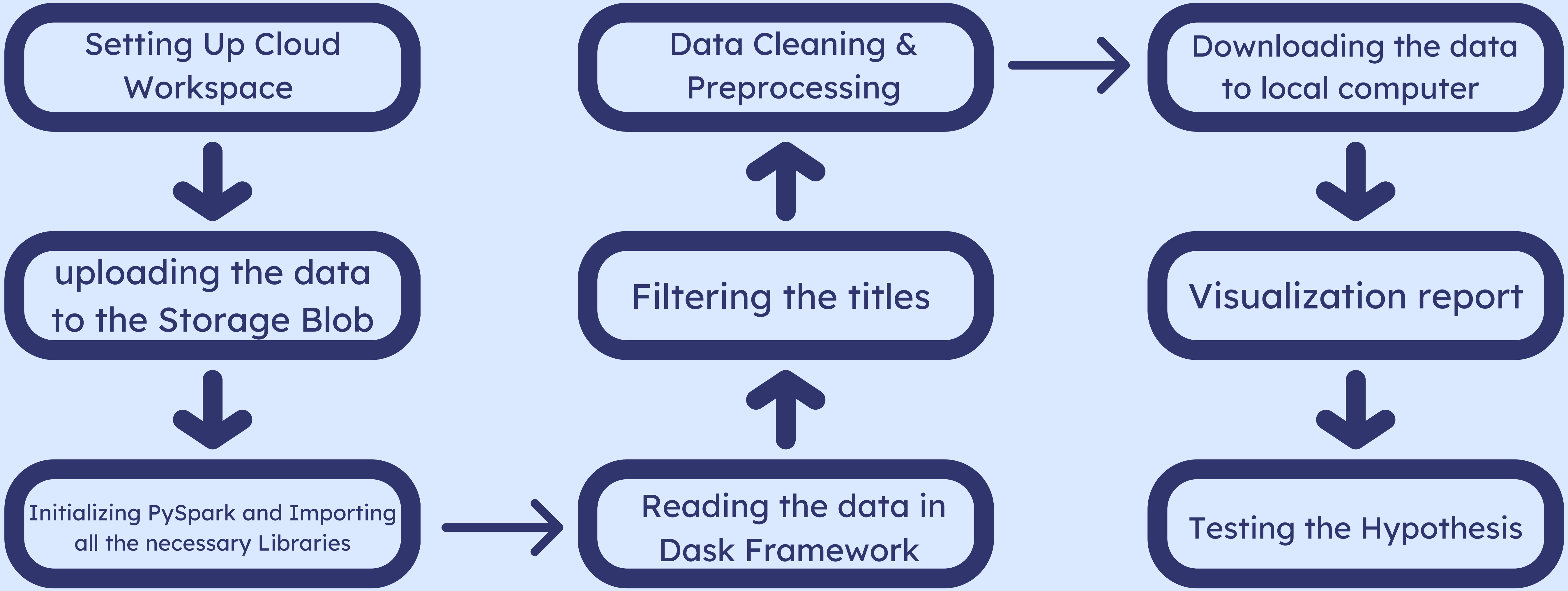
- Purpose: Provide actionable insights via a dynamic interface.

- Tools Used:
  - Visualization libraries: Power BI, Matplotlib.

Features:

- Skill trend analysis over time.
- Demand for programming languages and tools.
- Insights into geographical trends.
- Salary Impact over demanding Jobs.

Work Flow


Setting Up Cloud Workspace
uploading the data to the Storage Blob
Initializing PySpark and Importing all the necessary Libraries
Reading the data in Dask Framework
Filtering the titles
Data Cleaning & Preprocessing
Downloading the data to local computer
Visualization report
Testing the Hypothesis

Data Upload and Loading Process

Data Source:
The dataset was uploaded to the Kaggle environment for secure and scalable processing.
Library Used:
We utilized Dask DataFrame for handling the large dataset efficiently.

- Dask enables parallel and out-of-core computations, making it ideal for processing datasets that exceed memory capacity

In [3]:

```
# Load the CSV file with updated error handling
try:
    df = dd.read_csv(input_file, on_bad_lines='skip', engine='python')
    # Check the number of rows
    row_count = len(df)  # Get the number of rows
    print(f"Number of rows: {row_count}")
except Exception as e:
    print(f"An error occurred while reading the file: {e}")
```

```
Number of rows: 8852272
```

Data Transformation

Converting the CSV data to the Parquet format for optimized storage and querying:

```
# Save the filtered data to a Parquet file
filtered_df.to_parquet('/kaggle/working/filtered_full_data.parquet', engine='pyarrow', write_index=False)

print("Filtered dataset saved as Parquet: /kaggle/working/filtered_data.parquet")
```

Filtering and Preprocessing Data

Purpose of Filtering:

- To extract relevant job titles focusing on Big Data and related fields like:
  - Data Analyst, Big Data, Data Engineer, Machine Learning, Artificial Intelligence, etc.
- Reduces dataset size while keeping only meaningful data for analysis.

```python
# Define relevant keywords for filtering
keywords = [
    "Data Analyst", "Big Data", "Data Engineer", "Machine Learning",
    "Artificial Intelligence", "ETL", "Business Intelligence",
    "Cloud Data", "Data Scientist"
]

# Filter the DataFrame for job titles containing the keywords
filtered_df = df[df['title'].str.contains('|'.join(keywords), case=False, na=False)]

# Display the first few filtered results
print(filtered_df.head())

# Save the filtered results to a new Parquet file
filtered_df.to_parquet('/kaggle/working/filtered_titles.parquet', engine='pyarrow', write_
index=False)
```

```python
def extract_skills_debug(row):
    try:
        print(f"Raw row: {row}")  # Debugging: Print the raw row
        # Replace single quotes with double quotes and fix missing commas between dictionari
es
        formatted_row = re.sub(r'}\s*{', '}, {', row.replace("'", '"'))
        print(f"Formatted row: {formatted_row}")  # Debugging: Print the formatted row
        # Parse as JSON
        skills = json.loads(formatted_row)
        print(f"Parsed skills: {skills}")  # Debugging: Print the parsed JSON
        # Extract 'level' and 'name' for each skill
        skills_list = [{"level": skill.get("level"), "name": skill.get("name")} for skill
in skills]
        print(f"Skills list: {skills_list}")  # Debugging: Print extracted skills
        return pd.Series({
            "skills_names": ", ".join(skill["name"] for skill in skills_list if skill["nam
e"]),
            "skills_levels": ", ".join(str(skill["level"]) for skill in skills_list if ski
ll["level"]),
        })
    except Exception as e:
        print(f"Error processing row: {row} - {e}")
        return pd.Series({
            "skills_names": None,
            "skills_levels": None
        })
```

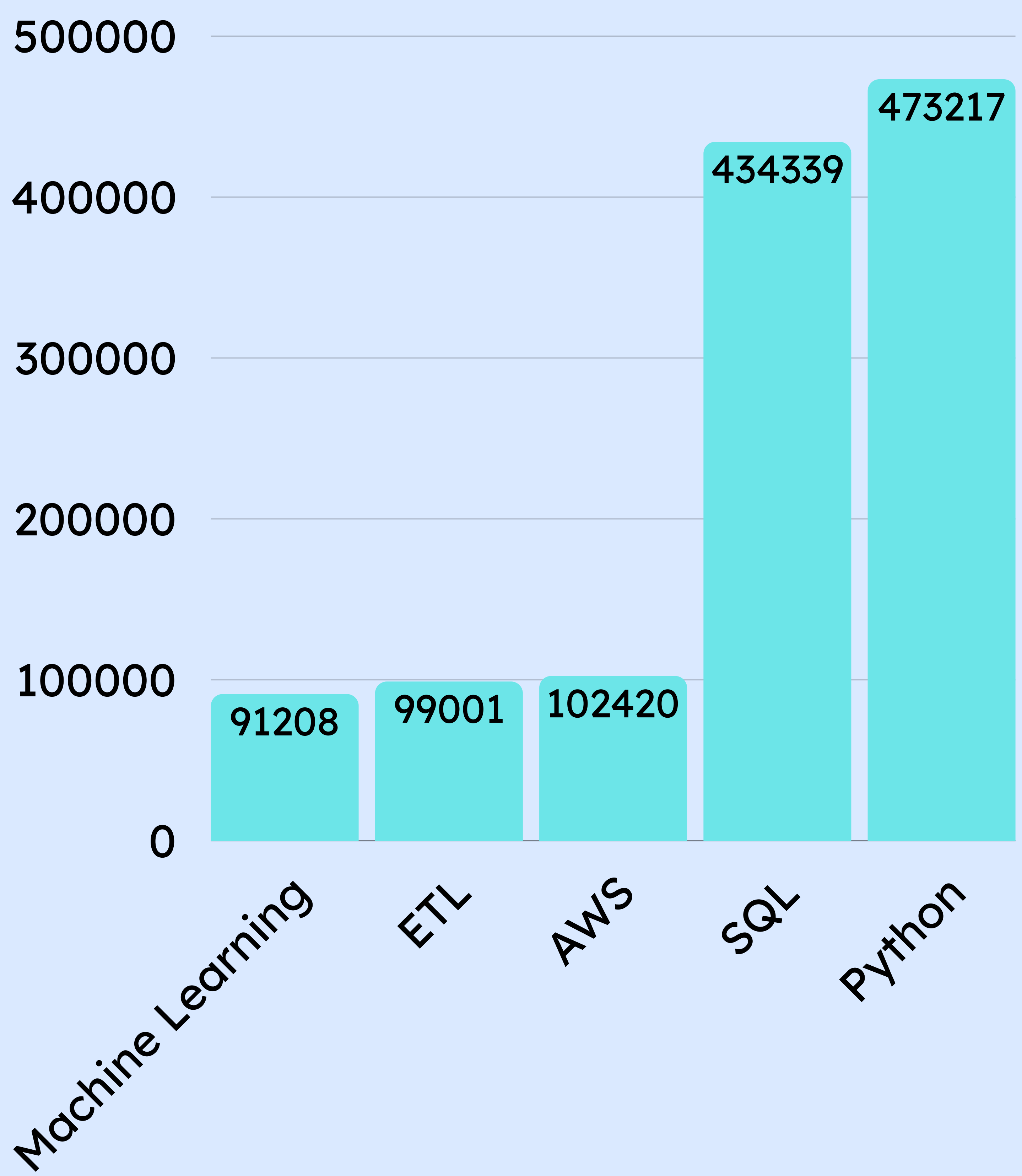
1- Salary preprocessing

2- Skills and Employment type Data Extraction

- Libraries Used:
  - ast for parsing nested structures in the employment_types column.
- Key function: ast.literal_eval() was utilized to parse JSON-like strings and extract salary details (salary_from, salary_to, currency) and employment types.

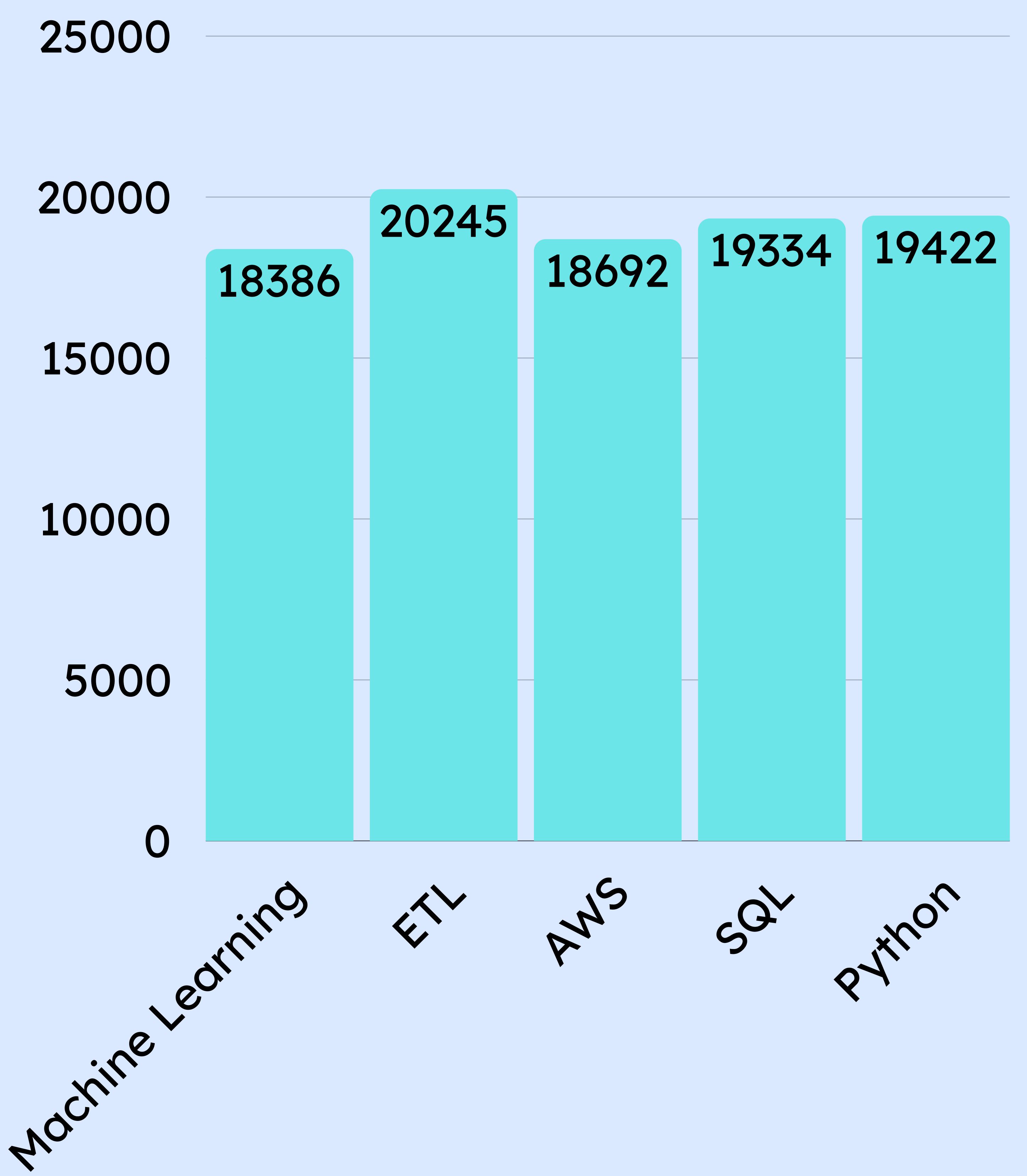
HYPOTHESIS 1

Skill Demand Hypothesis : Certain skills are higher in Demand in the IT Job Market



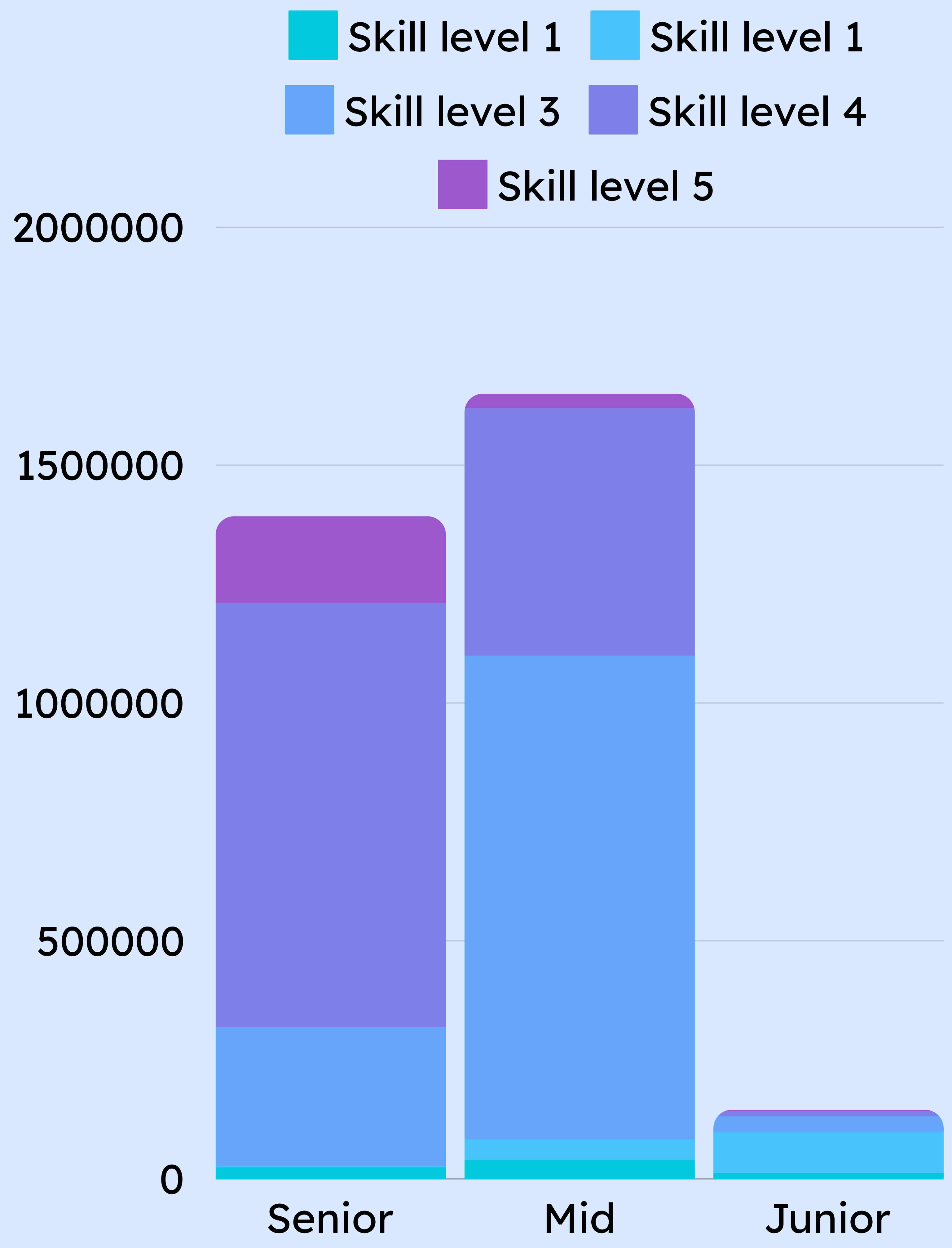
HYPOTHESIS 2

Salary Impact Hypothesis :
Job Requiring High Demand Skills tend to offer Higher salaries which Gives the insight “Peoples will be willing to pay for this course”



HYPOTHESIS 3

Skill Level Distribution Hypothesis: Senior-level job roles require more advanced skills compared to junior or mid-level roles.



HYPOTHESIS 4+5

4.	Geographic Demand Hypothesis: Demand for certain skills varies by location.
5.	Skill Trend Hypothesis: Some skills show a rising trend over time (e.g., ML frameworks or cloud computing).