My Approach:

Scraping LinkedIn violates its terms of service. We should use LinkedIn's API or obtaining consent before scraping.

I will use the "Selenium" library to log into LinkedIn with valid credentials and navigate to the LinkedIn search bar to find profiles matching a specific query, e.g., "IIT graduates". Extract relevant information (job titles, current companies, and industries) from the search results or individual profiles. handle dynamic content by waiting for elements to load and capturing all visible data.Store the extracted data in a structured format, such as a CSV file.We have to ensure that column headers are clear for later analysis.Handle challenges like CAPTCHA, network issues, or missing data.Use retry mechanisms for temporary issues and skip problematic profiles.Emphasize compliance with LinkedIn's terms of service and legal requirements.Recommend using LinkedIn's API for data access wherever possible.

The Python Script needed is:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
import csv


def scrape_linkedin(username, password, search_query):
  try:
      driver = webdriver.Chrome()
      driver.get("https://www.linkedin.com/login")
      time.sleep(3)
      driver.find_element(By.ID, "username").send_keys(username)
```

```
    driver.find_element(By.ID, "password").send_keys(password)
    driver.find_element(By.XPATH, "//button[@type='submit']").click()
    time.sleep(5)
    search_box = driver.find_element(By.XPATH, "//input[@placeholder='Search']")
    search_box.send_keys(search_query)
    search_box.send_keys(Keys.RETURN)
    time.sleep(5)
    profiles = []
    for _ in range(3):
        profile_cards = driver.find_elements(By.XPATH, "//div[contains(@class, 'search-result__info')]")
        for card in profile_cards:
            try:
                name = card.find_element(By.TAG_NAME, "span").text
                title = card.find_element(By.CLASS_NAME, "search-result__truncate").text
                company = card.find_element(By.CLASS_NAME, "subline-level-1").text
                industry = card.find_element(By.CLASS_NAME, "subline-level-2").text
                profiles.append({"Name": name, "Title": title, "Company": company, "Industry": industry})
            except Exception as e:
                print(f"Error parsing profile: {e}")
        next_button = driver.find_element(By.XPATH, "//button[contains(@aria-label, 'Next')]")
        next_button.click()
        time.sleep(5)
    with open("linkedin_profiles.csv", "w", newline="", encoding="utf-8") as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=["Name", "Title", "Company", "Industry"])
        writer.writeheader()
        writer.writerows(profiles)
    print("Data successfully scraped and saved to linkedin_profiles.csv.")
except Exception as e:
    print(f"Error: {e}")
finally:
    driver.quit()
```

```
username = "email"
password = "password"
search_query = "IIT graduates"
scrape_linkedin(username, password, search_query)
```

The script logs into LinkedIn using Selenium, entering credentials and bypassing the login screen.The script searches for IIT graduates, navigates through search results, and scrapes names, job titles, companies, and industries from visible profile cards.Saves the scraped data into a CSV file for easy analysis.Includes basic error handling for missing elements and network issues.

Implement reCAPTCHA solvers (e.g., 2Captcha) or manual intervention when CAPTCHA is detected.Use rotating proxies and user agents to avoid frequent CAPTCHA challenges.

Analyze the industries most IIT graduates enter, such as technology, consulting, or finance.Track job titles to study common career paths (e.g., software engineer product manager CTO).Identify the companies hiring the most IIT graduates.