* Data Scraping, Processing, and Visualization

Using Azure, GitHub, PowerBI &Tableau

* **Prepared by**:

Ahmed Abdelrahman  
Abdelrahman Hany  
Omar Elmnofy  
Mohammed Baligh

* **Supervised by**: Eng Khalid Seif

Table of Contents (Page 2)

1. Introduction
2. Data Scraping Overview
3. Scraping Workflow
4. Scraping Challenges
5. Azure Function for Data Upload
6. Pipeline 1 – Raw Data Ingestion
7. Pipeline 2 – Data Transformation and Warehousing
8. Data Modelling
9. PowerBI Dashboard Overview
10. PowerBI Visuals
11. Key Insights
12. Conclusion
13. Future Work
14. Challenges and Solutions
15. Team Credits

# 1. Introduction (Page 3)

* **Objective**: A comprehensive project focused on scraping data from Amazon’s regional websites (Egypt, India, Saudi Arabia, Japan), processing it using Azure Functions, and building pipelines for data transformation, warehousing, and visualization using PowerBI.
* **Motivation**: Explain why e-commerce data analysis is important and how cross-regional insights provide a competitive advantage.
* **Technologies Used**: Python (for scraping), Azure Functions, Data Pipelines, PowerBI.
* A screenshot of a computer

  Description automatically generated

# 2. Data Scraping Overview (Page 4)

* **Regions Covered**: Amazon Egypt, US, Saudi Arabia and Uk.
* **Data Collected**: Product name, price, availability, ratings, product category.
* **Scraping Tools**: Python libraries used:
  + **BeautifulSoup** for parsing HTML.
  + **Scrapy** for crawling and scraping.
  + **Selenium** for handling JavaScript-loaded content.

A screenshot of a computer

Description automatically generated

# 3. Scraping Workflow (Page 5)

* **Step-by-Step Process**:
  1. Sending requests to Amazon websites.
  2. Parsing HTML and extracting data.
  3. Cleaning and normalizing extracted data.
  4. Storing the data in CSV or JSON format.
* **Automation**: Discuss how Python scripts automate scraping, handle exceptions, and retry failed requests.

A screenshot of a computer

Description automatically generated

# 4. Scraping Challenges (Page 6)

* **Key Challenges**:
  + **CAPTCHAs**: Managed using proxies and delay handling.
  + **IP Blocking**: Solutions like rotating IP addresses.
  + **Dynamic Content**: Handled via Selenium for JavaScript-rendered pages.
* **Ethical Considerations**: Addressing Amazon’s scraping policies and ethical scraping practices.

# 5. Azure Function for Data Upload (Page 7)

* **Azure Function Overview**: Explain how serverless Azure Functions process scraped data.
* **Steps**:
  1. Receiving scraped data as JSON.
  2. Preprocessing and validating data.
  3. Storing data in **Azure Blob Storage** (Raw Data folder).
* **Scalability**: Benefits of using Azure Functions to automatically handle large-scale data uploads.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# 6. Pipeline 1 – Raw Data Ingestion (Page 8)

* **Pipeline Overview**: The first pipeline takes raw scraped data and ingests it into a **Raw Folder** in Azure.
* **Process**:
  1. Data ingestion from Azure Function.
  2. Storing and organizing raw data based on region and timestamp.
* **Tools**: Azure Data Factory for pipeline orchestration.

A screenshot of a computer

Description automatically generated

# 7. Pipeline 2 – Data Transformation and Warehousing (Page 9)

* **Data Transformation**:
  + Cleaning data (removing duplicates, handling missing values).
  + Normalizing currencies and product categories across regions.
* **Loading to Warehouse**: Transformed data loaded into an **Azure SQL Data Warehouse** for further analysis.
* **Azure Synapse**: Automating data flow and transformation.A screenshot of a computer

  Description automatically generated

# 8. Data Modeling (Page 10)

* **Star Schema Design**:
  + **Fact Table**: Storing metrics (e.g., sales data, prices).
  + **Dimension Tables**: Product details, regions, time-based information.A screenshot of a computer

    Description automatically generated
* **Why Star Schema?**: Efficient querying and easy-to-use structure for PowerBI reports.

A screenshot of a computer

Description automatically generated

* A computer screen shot of a computer screen

  Description automatically generated

# 9. PowerBI Dashboard Overview (Page 11)

* **Connecting PowerBI to Data Warehouse**: Steps for setting up PowerBI with the data warehouse.
* **Dashboard Layout**: Overview of dashboard features:
  + Filters for region, time, and category.
  + Interactive visual elements (graphs, charts).A screenshot of a computer

    Description automatically generated

# 10. PowerBI Visuals (Page 12)

* **Key Visualizations**:
  + **Price Trends**: Line graphs showing price variations over time across regions.
  + **Top Products**: Bar charts showing the most popular products in each region.
  + **Regional Comparison**: Maps visualizing product availability and pricing in different regions.

A screenshot of a computer

Description automatically generated

## 11. Key Insights (Page 13)

* **Insights Gained**:
  + **Regional Price Variation**: Notable price differences for the same product across different regions.
  + **Best-Selling Products**: Insights into product popularity based on region.
  + **Seasonality**: Seasonal fluctuations in prices and availability during holidays or sales events.

A screenshot of a computer

Description automatically generated

## 12. Conclusion (Page 14)

* **Project Summary**:
  + Successfully built a multi-regional data scraping pipeline.
  + Automated data ingestion, transformation, and storage using Azure services.
  + Visualized key insights using PowerBI, providing valuable e-commerce intelligence.
* **Key Takeaways**: Discuss the technical and business advantages provided by the project.

## 13. Future Work (Page 15)

* **Suggestions for Expansion**:
  + **Additional Regions**: Scraping Amazon websites from other countries (e.g., US, UK).
  + **Real-Time Data**: Automating real-time scraping and pipeline execution for continuous updates.
  + **Machine Learning Integration**: Predictive analytics to forecast pricing trends or demand spikes.

## 14. Challenges and Solutions (Page 16)

* **Challenges Encountered**:
  + Handling large volumes of data.
  + Ensuring data consistency across regions.
* **Solutions**:
  + Using parallel processing for faster scraping and transformations.
  + Implementing data validation scripts to ensure quality.

## 15. Team Credits (Page 17)

* **Team Members**:
  + Ahmed Abdelrahman
  + Abdelrahman Hany
  + Omar Elmnofy
  + Mohammed Baligh
* **Supervised by**: Eng Khalid Seif

# References (Page 18)

* **List of tools, documentation, libraries, and research used during the project.**
* **Scraping Tools and Libraries**
* **Python:** A popular language for web scraping due to its versatility and extensive libraries.
* **Beautiful Soup:** A Python library for parsing HTML and XML documents.
* **Scrapy:** A powerful framework for web scraping that can handle large-scale projects.
* **Selenium:** A browser automation tool that can be used for dynamic websites.
* **Node.js:** Another popular language for web scraping with libraries like Puppeteer.
* **R:** A statistical programming language with libraries like rvest for web scraping.
* **References and Documentation**
* **Amazon's Terms of Service:** Ensure you comply with Amazon's guidelines for web scraping.
* **Web Scraping Best Practices:** Learn about ethical considerations, avoiding detection, and handling rate limits.
* **Python Web Scraping Tutorial:** A comprehensive guide to using Python for web scraping.
* **Scrapy Documentation:** Official documentation for the Scrapy framework.
* **Selenium Documentation:** Official documentation for the Selenium browser automation tool.
* **Research**
* **Amazon's Website Structure:** Understand how Amazon's websites are structured to identify the elements you need to scrape.
* **Dynamic Content:** If Amazon uses dynamic content, explore techniques like JavaScript rendering or headless browsers to extract data.
* **Rate Limiting:** Research how to avoid being blocked by Amazon's rate limits.
* **Data Cleaning and Normalization:** Learn about techniques to clean and normalize the scraped data before uploading.
* **Uploading to Azure Function**
* **Azure Functions Documentation:** Refer to Azure Functions documentation for guidance on creating and deploying functions.
* **Python or Node.js:** Use Python or Node.js to write your Azure Function.
* **HTTP Trigger:** Configure your function to be triggered by an HTTP request.
* **Data Pipeline**
* **Azure Data Factory:** A powerful tool for creating and managing data pipelines.
* **Azure Data Factory Documentation:** Refer to Azure Data Factory documentation for guidance on creating pipelines.
* **Data Transformation and Warehousing:** Use Azure Data Factory to transform the data and load it into a data warehouse or data lake.
* **Data Modelling:** Design a data model that reflects your business requirements.
* **Presenting Data with Power BI**
* **Power BI Documentation:** Refer to Power BI documentation for guidance on creating reports and dashboards.
* **Data Visualization:** Learn about best practices for data visualization to effectively present your insights.
* **Connecting to Azure Data Warehouse:** Connect Power BI to your Azure data warehouse to visualize the data.
* **Additional Considerations:**
* **Data Quality:** Ensure the quality of your scraped data by implementing validation and cleaning processes.
* **Scalability:** If you're dealing with large amounts of data, consider using scalable solutions like Azure Data Factory and Azure Data Lake Storage.
* **Security:** Protect your data by implementing appropriate security measures.
* **Legal and Ethical Considerations:** Adhere to legal and ethical guidelines when scraping data.