

# systems

## **Internship Report**

Prepared by: Usama Yazdani

Submitted to : Sir Matteen Sajjad Daar

Date : 10/18/2024

## Table of Contents:

### Project 1: Microsoft Fabric - Bing News Data Analytics (End-to-End Azure Data Engineering Project)

**1: Introduction**

2: Data Gathering Process

**3: Data Storage**

**4: Data Processing**

**5: Data Warehouse Setup**

**6: Semantic Model Creation**

**7: Data Visualization**

**9: Automation**

10: Additional Ingestions

### Project 2: Power BI - Data Professional Survey Breakdown

1:Introduction

**2: Data Source and Fields**

**3: Visualizations**

**4: Insights**

**5: Conclusion**

# Project 1: Bing News Data Analytics - End-to-End Azure Data Engineering using Microsoft Fabric

## Overview Of Microsoft fabric

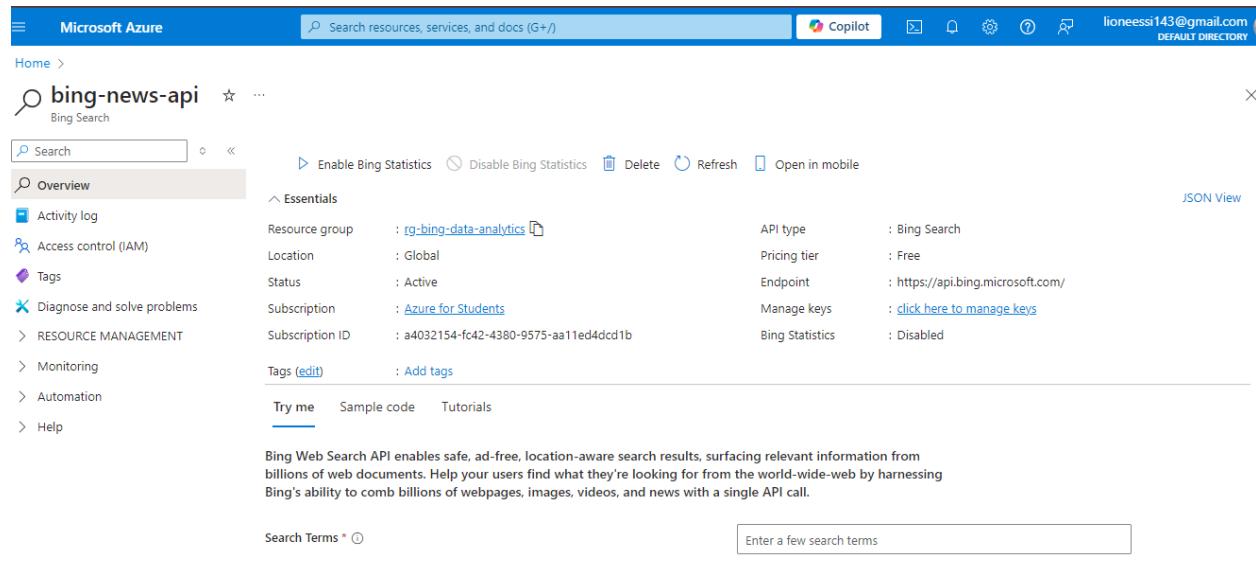
Microsoft Fabric is a comprehensive, unified analytics platform that brings together Azure Data Factory, Power BI, Synapse Analytics, and data lakes, making it an ideal solution for end-to-end data engineering and analytics workflows. It provides a single pane of glass for all data operations, allowing seamless integration between data ingestion, transformation, storage, and visualization.

## Step 1: Data Gathering

To begin the project, I used **Microsoft Azure Marketplace** to source data. The marketplace offers a free **Bing News API**, which I used to retrieve real-time news articles. I connected to the API and configured it to gather news based on specific topics and keywords.

**API Configuration:** Used the Bing News API from Azure Marketplace, filtering data for relevant news categories.

**Data Storage Format:** The retrieved data was stored in JSON format to ensure flexibility and ease of transformation.



The screenshot shows the Microsoft Azure portal interface. At the top, there's a blue header bar with the Microsoft Azure logo, a search bar, and various navigation icons. The main content area has a white background. On the left, there's a sidebar with a tree view showing the resource group 'rg-bing-data-analytics' and several sub-options like 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'RESOURCE MANAGEMENT', 'Monitoring', 'Automation', and 'Help'. The main panel displays the 'bing-news-api' resource. It has a title bar with 'bing-news-api' and a 'Bing Search' icon. Below the title bar are several buttons: 'Search' (with a magnifying glass icon), 'Enable Bing Statistics', 'Disable Bing Statistics', 'Delete', 'Refresh', and 'Open in mobile'. To the right of these buttons is a 'JSON View' link. The main content area is titled 'overview' and contains a table with the following data:

Essentials	
Resource group	: rg-bing-data-analytics
Location	: Global
Status	: Active
Subscription	: Azure for Students
Subscription ID	: a4032154-fc42-4380-9575-aa11ed4dc1b
Tags (edit)	: Add tags

Below the table, there are links for 'Try me', 'Sample code', and 'Tutorials'. At the bottom of the main panel, there's a note about the Bing Web Search API and a search bar at the bottom right labeled 'Enter a few search terms'.

## Step 2: Data Ingestion and Storage in Lakehouse

After fetching the data via the API, I utilized **Microsoft Fabric's Data Factory** to load the data into a **Lakehouse**. The Data Factory allowed for smooth orchestration of the pipeline, ensuring that the data was ingested in real-time.

- **Lakehouse Creation:** The JSON files were stored in **Microsoft Fabric's Lakehouse**, which served as the primary data storage location.
- **Data Organization:** The Lakehouse enabled structured storage, making it easier to perform downstream transformations.

The image displays two screenshots of Microsoft Fabric interfaces. The top screenshot shows the Data Factory pipeline 'news-ingestion-pipeline' with a 'Copy data' step mapping 'bing-news-search-connection 00...' to a 'Notebook' named 'Data Transformation'. The bottom screenshot shows the OneLake interface for the lakehouse 'bing\_lake\_db', where a file named 'bing-latest-news.json' has been uploaded to the 'tbl\_latest\_news' table.

**Top Screenshot: Microsoft Fabric Data Factory Pipeline**

The pipeline 'news-ingestion-pipeline' is shown with the 'Source' tab selected. The 'Connection' dropdown is set to 'bing-news-search-connection 00...'. The 'Relative URL' field contains 'https://api.bing.microsoft.com/v7.0/'. A red box highlights this section.

**Bottom Screenshot: Microsoft Fabric OneLake Interface**

The lakehouse 'bing\_lake\_db' is shown. The 'Explorer' pane shows a tree structure with 'bing\_lake\_db' expanded, revealing 'Tables' and 'Files'. Under 'Tables', 'tbl\_latest\_news' is selected. The 'Files' pane lists several files, with 'bing-latest-news.json' highlighted by a red box. The file details are as follows:

Name	Date modified	Type	Size
MT cars.parquet	9/10/2024 1:12:...	parquet	2 KB
POWER_Regional_Monthly_1984_2022.csv	9/10/2024 1:12:...	csv	299 KB
POWER_Regional_Monthly_2010_2022.csv	9/10/2024 1:12:...	csv	68 KB
POWER_Regional_Monthly_2020_2022.csv	9/10/2024 1:12:...	csv	62 KB
<b>bing-latest-news.json</b>	9/10/2024 1:24:...	json	10 KB

## Step 3: Data Preprocessing using Synapse Spark Notebooks

Once the data was stored in the Lakehouse, I used **Azure Synapse Data Engineering** to preprocess and clean the data. This was done through a **Synapse Spark Notebook** where I implemented the necessary transformations in **Python** using Spark.

- **Data Cleaning:** Removed any irrelevant fields, handled missing data, and optimized the structure for further processing.
- **Preprocessing:** Applied transformations such as filtering based on certain criteria (e.g., news date range) and formatting fields.
- Design a schema : I design a schema related to news and also create a view there so the new news should not overwrite the existing one.

The image shows two screenshots of the Power BI workspace interface, each displaying a notebook titled "processing-bing-news".

**Screenshot 1:** This screenshot shows the "PySpark (Python)" tab selected. The code in the editor is as follows:

```

1 #convert different item into json dictionary and load them into list
2 import json
3
4 title = []
5 description = []
6 category = []
7 url = []
8 image = []
9 provider = []
10 datePublished = []
11
12 for counter_str in json_list:
13     try:
14         news_json = json.loads(counter_str) #load json into dictionary
15         # no processing if the json from api does not contain category or image
16         if(not news_json['json_object'].get('category') is None and not news_json['json_object'].get('image')):
17             title.append(news_json['json_object']['name'])
18             description.append(news_json['json_object']['description'])
19             category.append(news_json['json_object']['category'])
20             url.append(news_json['json_object']['url'])
21             image.append(news_json['json_object']['image']['thumbnail']['contentUrl'])
22             provider.append(news_json['json_object']['provider']['name'])

```

**Screenshot 2:** This screenshot also shows the "PySpark (Python)" tab selected. The code in the editor is as follows:

```

4 data = list(zip(title,description,category,url,image,provider,datePublished))
5
6 schema = StructType([
7     StructField("title", StringType(), True),
8     StructField("description", StringType(), True),
9     StructField("category", StringType(), True),
10    StructField("url", StringType(), True),
11    StructField("image", StringType(), True),
12    StructField("provider", StringType(), True),
13    StructField("datePublished", StringType(), True)
14 ])
15
16 df_cleaned = spark.createDataFrame(data,schema =schema)

```

Below this, another command is shown:

```

1   from pyspark.sql.functions import to_date, date_format
2   df_cleaned_final = df_cleaned.withColumn("datePublished", date_format(to_date("datePublished"), "dd-MM-yyyy"))

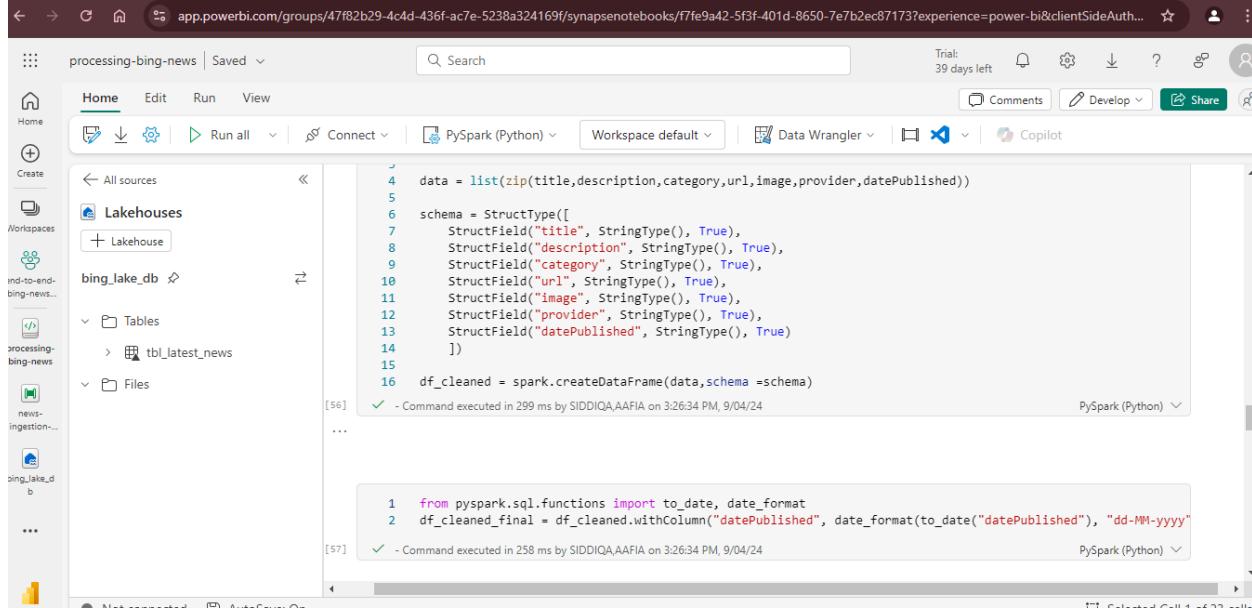
```

## Step 4: Data Modeling – Creating Dimensions and Fact Tables

After preprocessing, the next step was to model the data in a structured form, suitable for reporting. I used **T-SQL** to define **dimension tables** and a **fact table**, ensuring the data was ready for analysis in the warehouse.

- **Fact Table:** Created a fact table that captures the key metrics (e.g., news image, Url, etc.).

- **Dimension Tables:** Established dimension tables to categorize data (e.g., Categories, Provider, etc.).
- **Warehouse:** All the data was then loaded into a **Data Warehouse** using Microsoft Fabric.

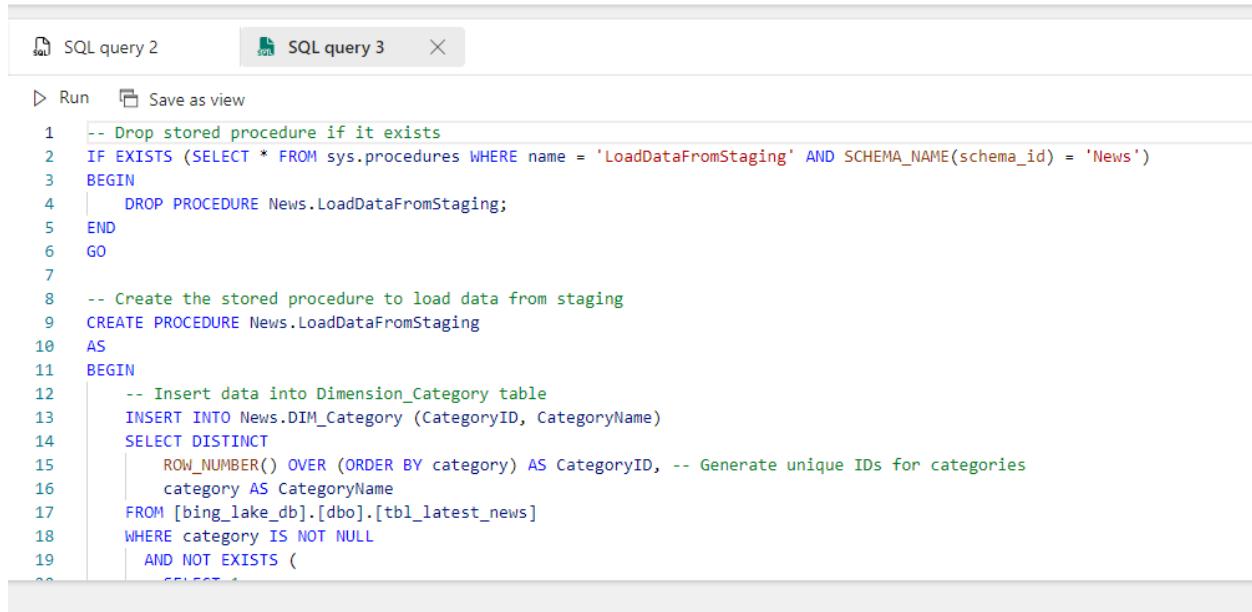


```

4 data = list(zip(title,description,category,url,image,provider,datePublished))
5
6 schema = StructType([
7     StructField("title", StringType(), True),
8     StructField("description", StringType(), True),
9     StructField("category", StringType(), True),
10    StructField("url", StringType(), True),
11    StructField("image", StringType(), True),
12    StructField("provider", StringType(), True),
13    StructField("datePublished", StringType(), True)
14 ])
15
16 df_cleaned = spark.createDataFrame(data,schema =schema)
17
18 from pyspark.sql.functions import to_date, date_format
19 df_cleaned_final = df_cleaned.withColumn("datePublished", date_format(to_date("datePublished"), "dd-MM-yyyy"))

```

After creating a table then insert data from lakehouse to warehouse :



```

1 -- Drop stored procedure if it exists
2 IF EXISTS (SELECT * FROM sys.procedures WHERE name = 'LoadDataFromStaging' AND SCHEMA_NAME(schema_id) = 'News')
3 BEGIN
4     DROP PROCEDURE News.LoadDataFromStaging;
5 END
6 GO
7
8 -- Create the stored procedure to load data from staging
9 CREATE PROCEDURE News.LoadDataFromStaging
10 AS
11 BEGIN
12     -- Insert data into Dimension_Category table
13     INSERT INTO News.DIM_Category (CategoryID, CategoryName)
14     SELECT DISTINCT
15         ROW_NUMBER() OVER (ORDER BY category) AS CategoryID, -- Generate unique IDs for categories
16         category AS CategoryName
17     FROM [bing_lake_db].[dbo].[tbl_latest_news]
18     WHERE category IS NOT NULL
19     AND NOT EXISTS (
20         SELECT 1
21     )
22 
```

And running query on it :

The screenshot shows the Microsoft Data Studio interface. At the top, there's a navigation bar with 'bing\_news\_warehouse' selected, a search bar, and a trial status message 'Trial: 39 days left'. Below the navigation bar is a toolbar with icons for 'Get data', 'New SQL query', 'New visual query', 'Query activity', and 'Download SQL database project'. The main area has tabs for 'SQL query 2', 'SQL query 3', 'SQL query 4', and 'SQL query 5'. 'SQL query 5' is active and contains the following SQL code:

```

1  SELECT * FROM News.FACT_news;
2  select * from News.DIM_Category;
3  SELECT * from News.DIM_date;
4  SELECT * from News.DIM_provider;
5
6  select p.ProviderName FROM News.DIM_provider as p inner join News.FACT_news as F
7  on p.ProviderID=F.ProviderID INNER JOIN News.DIM_date as D on D.DateID=F.DateID WHERE D.Day=2;
8
9

```

The 'Results' tab is selected in the results viewer, which displays the output of the last query:

	ABC ProviderName
1	deaandeelhouder.nl

## Step 5: Creating the Semantic Model and Power BI Report

To visualize the data, I built a **semantic model** that defined relationships between fact and dimension tables, making it easier to interpret the data. Afterward, I connected this semantic model to **Power BI** for reporting.

- **Semantic Model:** Created a model that mapped the relationships between the datasets.
- **Power BI Dashboard:** Used Power BI to generate insightful reports, including trending news topics, sentiment analysis, and category breakdowns.

The screenshot shows a Power BI dashboard titled "News Dashboard". At the top left, there's a dropdown menu set to "news\_dashboard" and a search bar. The top navigation bar includes "File", "View", "Reading view", "Mobile layout", "Open data model", and a Microsoft Fabric logo. On the left, there's a filter pane for "datePublished" with a dropdown showing "All". To the right, a large bold "2" is displayed with the text "Count of news" underneath. The main content area displays a table with two rows of news items. The columns are "title", "provider", "url", "category", and "datePublished". The first row shows "Atos past financiële verwachtingen aan" from "Knack" with the URL <https://datanews.knack.be/nieuws/bedrijven/financieel/atos-past-financiële-verwachtingen-aan/>, categorized as "Business" and published on "03-09-2024". The second row shows "Update: Atos rekent op minder omzet" from "desandeelhouder.nl" with the URL <https://www.desandeelhouder.nl/nieuws/2024/09/02/update-atos-rekent-op-minder-omzet/>, also categorized as "Business" and published on "02-09-2024".

title	provider	url	category	datePublished
Atos past financiële verwachtingen aan	Knack	<a href="https://datanews.knack.be/nieuws/bedrijven/financieel/atos-past-financiële-verwachtingen-aan/">https://datanews.knack.be/nieuws/bedrijven/financieel/atos-past-financiële-verwachtingen-aan/</a>	Business	03-09-2024
Update: Atos rekent op minder omzet	desandeelhouder.nl	<a href="https://www.desandeelhouder.nl/nieuws/2024/09/02/update-atos-rekent-op-minder-omzet/">https://www.desandeelhouder.nl/nieuws/2024/09/02/update-atos-rekent-op-minder-omzet/</a>	Business	02-09-2024

## Step 6: Automation and Orchestration

To ensure the data was always up-to-date, I implemented an **automated orchestration pipeline** using **Microsoft Fabric's Data Factory**. This pipeline ran every **1 hour** to fetch the latest news data from the API and update the existing reports and dashboards.

- **Orchestration:** The process was automated to run at regular intervals, ensuring that the latest news data was always available for analysis.
- **Scheduling:** Implemented a trigger that initiates the pipeline every hour to update the warehouse and refresh the Power BI reports.

The screenshot shows the Microsoft Fabric interface for managing a data pipeline named "news-ingestion-pipeline". The "Schedule" tab is active. Key settings include:

- Schedule**: A link to manage scheduled runs.
- Scheduled run**: A switch set to **On**.
- Repeat**: Set to **Daily**.
- Time**: Set to **01:24 PM**, with a trash icon for deletion.
- Add a time**: A button to add more times.
- Start date and time**: Set to **09/05/2024**.
- End date and time**: Set to **09/10/2024**.
- Time zone**: Set to **(UTC+05:00) Islamabad, Karachi**.

-----End -----

## Additional Tasks in Microsoft Fabric

Besides the Bing News project, I worked on a few other tasks assigned by my manager, including:

### 1. Data Ingestion from Snowflake:

- I ingested data from Snowflake into Microsoft Fabric using the **Data Factory**, integrating Snowflake's structured data into the Lakehouse for further processing.

The screenshot shows the Microsoft Fabric Data Flow interface. At the top, there's a navigation bar with 'Home', 'Activities', 'Run', 'View', and various icons for validation, scheduling, triggers, run history, copy data, dataflow, and search. A trial status 'Trial: 39 days left' is also visible. Below the navigation is a toolbar with icons for file, edit, settings, validate, run, schedule, trigger, view run history, copy data, dataflow, and search.

The main area displays a 'Copy data1' activity card. The 'Source' tab is selected. The configuration pane shows:

- Connection:** oa59927.ap-south-1.aws.snowflake... (selected)
- Database:** TEST
- Use query:** Table (selected)
- Table:** TEST1/CARS

Below the configuration pane, there are buttons for Refresh, Test connection, Edit, Preview data, and Enter manually.

## Data Ingestion from AWS S3:

- Similarly, I fetched data from **AWS S3 buckets** and ingested it into the Microsoft Fabric environment for analysis.

The screenshot shows the Microsoft Fabric Data Flow interface, similar to the previous one but for AWS S3. The 'Source' tab is selected in the configuration pane. The configuration pane shows:

- Connection:** https://s3.amazonaws.com 00000... (selected)
- Connection type:** Amazon S3
- File path type:** Wildcard file path (selected)
- Bucket:** juststoragebucket
- Wildcard paths:** Wildcard folder path / \*
- Recursively:** (checkbox checked)
- File format:** Binary

Below the configuration pane, there are buttons for Refresh, Test connection, Browse, Preview data, and Settings.

## Conclusion and Final Notes

Through this project, I gained hands-on experience with **end-to-end data engineering** on Microsoft Fabric, working with tools like Azure Synapse, Power BI, and automated orchestration. The integration of real-time data sources and building of reusable data models provided valuable insights, and the automation ensured that the reports were consistently up-to-date without manual intervention.

# Power BI - Data Professional

## Survey Breakdown

### ***Introduction:***

In this project, I work on survey dataset which is related to people job and their related things. I used power bi to create a interactive dashboard where put can have multiple functionality to do analysis.

The focus of the analysis was to break down key insights like demographics, salary trends, job satisfaction, and programming preferences.

### ***. Data Source:***

- The dataset contained fields like job titles, salary, country of origin, favorite programming languages, and happiness metrics.

Data Professional Survey	
Σ	Average Salary
>	Date Taken (America/New_York)
	Email
	Q1 - Which Title Best Fits your Current Role?.1
Σ	Q10 - Current Age
	Q11 - Which Country do you live in?.1
	Q12 - Highest Level of Education
	Q13 - Ethnicity.1
	Q2 - Did you switch careers into Data?
	Q3 - Current Yearly Salary (in USD)
	Q4 - What Industry do you work in?.1
	Q5 - Favorite Programming Language.1
Σ	Q6 - How Happy are you in your Current Position with the follo...
Σ	Q6 - How Happy are you in your Current Position with the follo...
Σ	Q6 - How Happy are you in your Current Position with the follo...
Σ	Q6 - How Happy are you in your Current Position with the follo...
Σ	Q6 - How Happy are you in your Current Position with the follo...
Σ	Q6 - How Happy are you in your Current Position with the follo...

## Key Metrics and Visualizations:

- **Count of Survey Takers**
- **Count of Survey Takers**
- **Favorite Programming Languages:**
- **Difficulty to Break into the Data Profession:**
- **Happiness Metrics:**

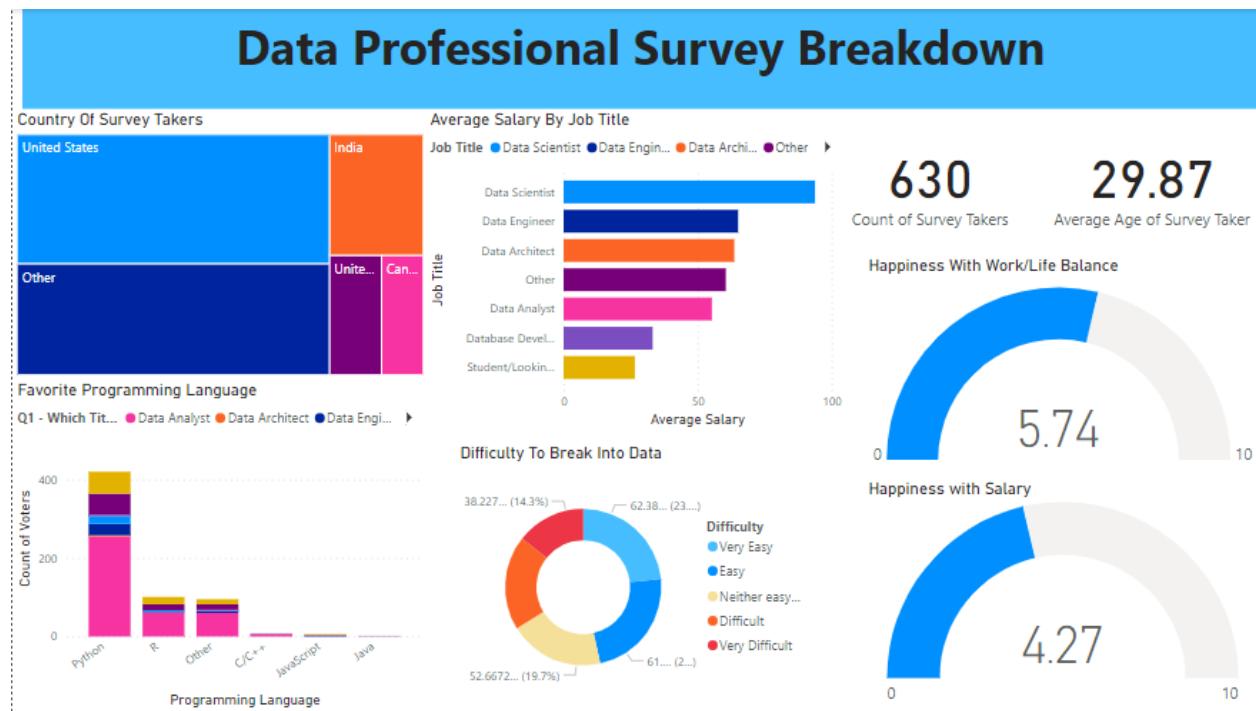
## Insight :

- U.S. and India dominate the data profession in terms of participation.
- Data Scientists tend to have higher average salaries than Data Engineers.
- Python remains the top programming language across most job roles.
- A significant number of professionals find it moderately difficult to break into the field.

- Overall happiness with salary is moderate, with work/life balance ratings being relatively higher.

## Conclusion:

State that this analysis can help organizations or job seekers get an overview of what to expect in terms of salary, challenges, and satisfaction within the industry.



> Thank You 😊 <