
Data Lake Analytics

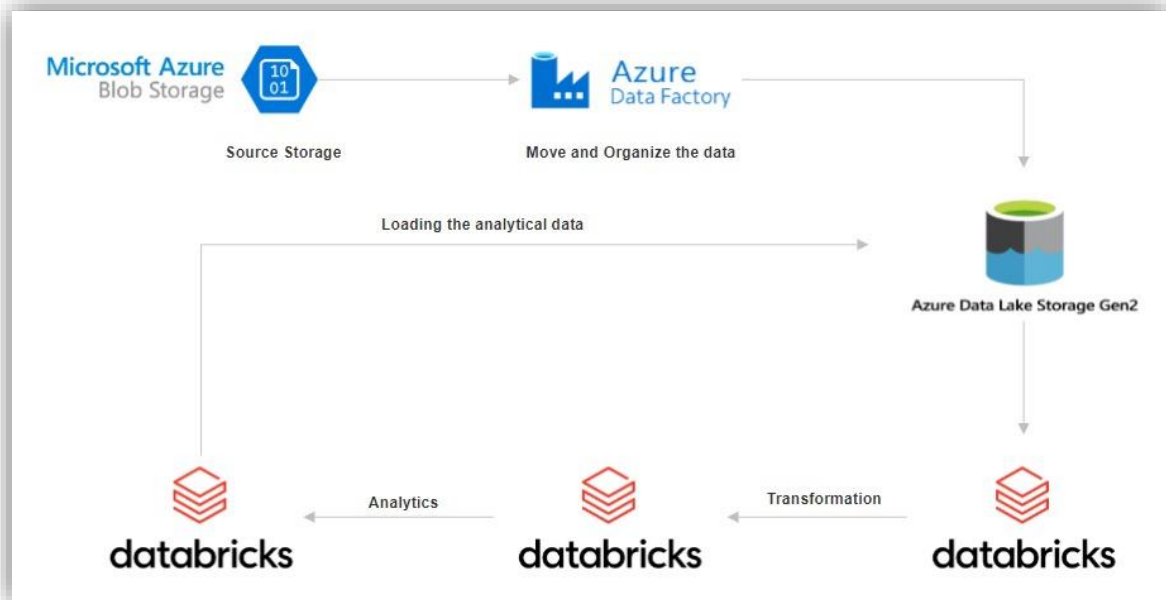
Project Overview

In today's data-driven world, the ability to efficiently process and analyse large volumes of data is crucial for businesses to gain insights and make informed decisions. This project aims to leverage the power of Azure Databricks and PySpark to perform data analytical tasks, including Extract, Transform, and Load (ETL) operations, on massive datasets.

About Project

This project revolves around the seamless orchestration of data movement, transformation, and analytics by integrating Azure Data Factory (ADF) and Azure Databricks. The workflow involves moving data from Azure Blob Storage to Azure Data Lake Storage Gen2 (ADLS Gen2) using ADF, mounting the ADLS Gen2 within Databricks notebooks for data transformations and analytics, and finally, persisting the analytical results back to ADLS Gen2.

Architectural Diagram



Key-Components/Requirements of the projects

1. Azure Databricks:

- Azure Databricks provides a cloud-based platform for big data analytics and machine learning. It offers a collaborative environment for data engineers, data scientists, and analysts to work together seamlessly.
- Databricks provides managed Spark clusters, eliminating the need for infrastructure management and allowing teams to focus on data processing tasks.

2. PySpark:

- PySpark is the Python API for Apache Spark, a powerful open-source framework for distributed data processing. PySpark simplifies development tasks by providing a Python interface to Spark's capabilities.
- With PySpark, developers can write concise and expressive code to perform complex data transformations, aggregations, and analytics on large datasets.

3. Azure Data Factory:

- Data Factory (ADF) allows users to create, schedule, and manage data pipelines that can move data between various supported data stores. ADF provides a scalable, fully managed platform for orchestrating and automating data workflows.
- It has many features such as data orchestration, seamless integration with Azure services, hybrid data integration, security, scalability, monitoring, metadata management, and cost management, enabling users to create, automate, and manage data pipelines for diverse data workflows.

Azure Resources Used for this Project

1. Azure Blob Storage

- This is where the raw data is stored. Azure Blob Storage integral to Microsoft Azure's storage service, is a cloud-based solution tailored for managing vast amounts of unstructured data, encompassing both text and binary data. Termed "Blob" for "Binary Large Object," it signifies a compilation of binary data treated as a singular entity within a database.

2. Azure Data Factory

- Here, we use azure data factory for moving and organizing the data from blob storage to azure data lake gen 2. It provides essential tools to create pipelines for the movement of data.
- It also provides Cloud-based service for orchestrating, automating data workflows, enabling seamless movement, transformation, and integration across diverse sources.

3. Azure Data Lake Storage Gen2

- This is where the raw data is loaded by ADF and then the analytical data is also moved and organized here. Azure Data Lake Storage Gen2 provides a scalable and secure platform for storing large volumes of data. It enables us to manage, access, and analyse data effectively.

4. Azure Databricks

- Azure Databricks are used to develop essential notebooks that contains the pyspark code which perform transformation and analytical operations on the data based on the business requirement.

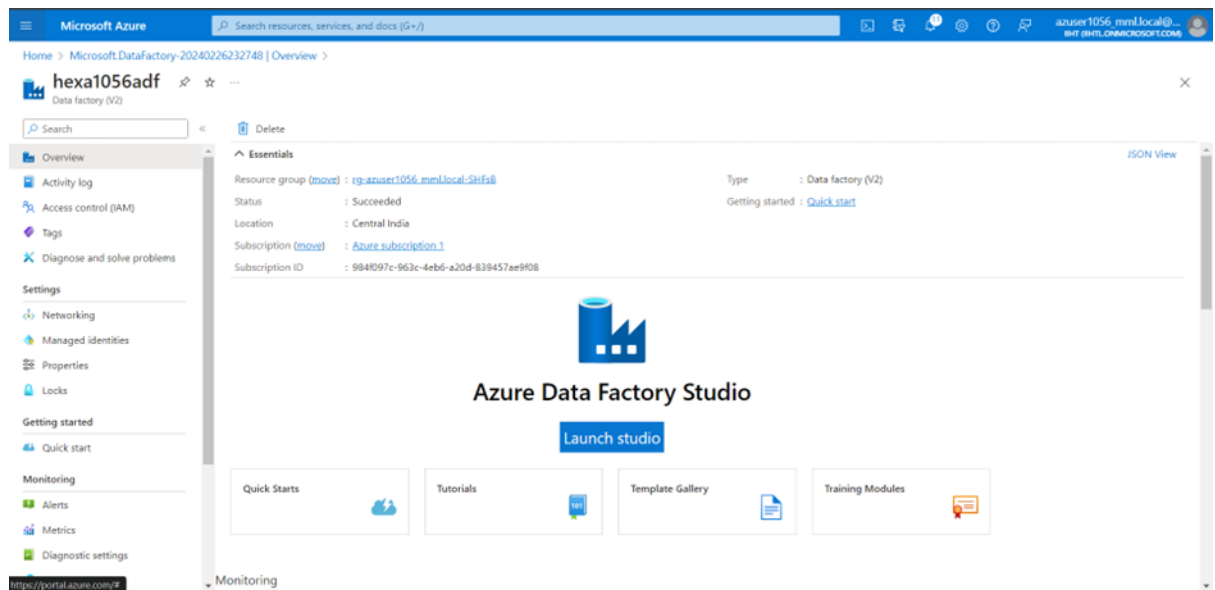
5. Databricks Cluster

- An Azure Databricks cluster process the data depending on the user instructions in the Azure Notebook. It serves as a computational resource facilitating the processing of extensive data and execution of analytics workloads through the Apache Spark platform within the Microsoft Azure cloud.
- Azure Databricks are used to develop essential notebooks that contains the pyspark code which perform transformation and analytical operations on the data based on the business requirement.

How It works

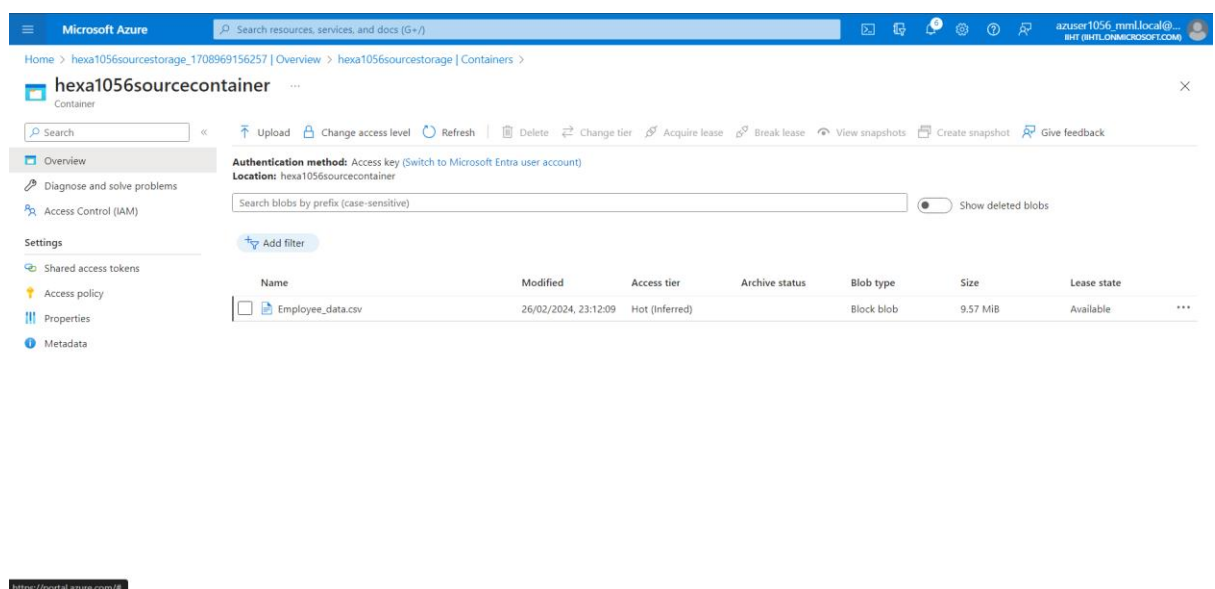
1. Setting Up Azure Databricks Factory:

- Sign in to the Azure portal and create an Azure Data Factory. Configure the settings, including pricing tier, region, and workspace name.

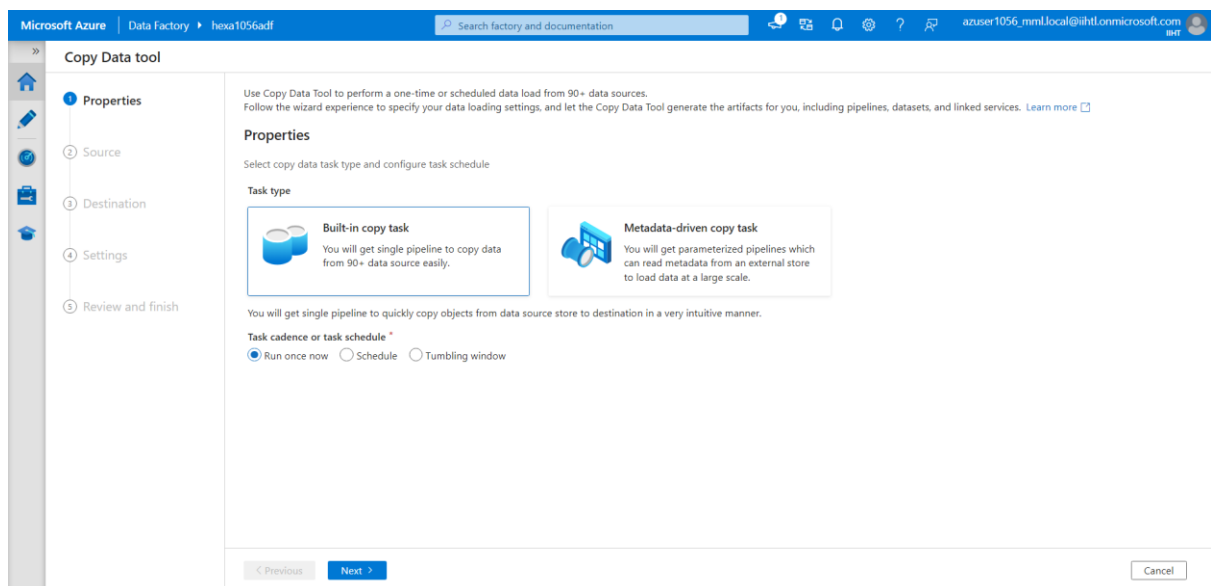
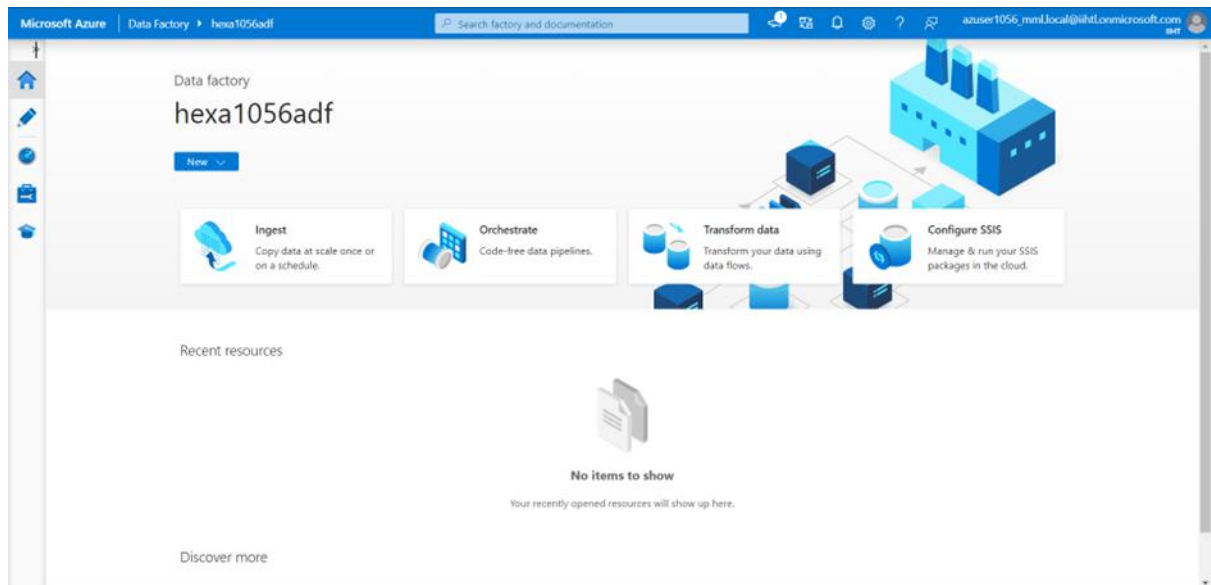


2. Movement of data:

- Raw data is stored in source storage (Azure Blob Storage)



- Ingest the data from the source storage(blob storage) to sink storage(Azure data lake gen 2) by creating a pipeline in the azure data factory with the following properties.



- Create the connection with the source storage and mention the right file path of the raw data.

Microsoft Azure | Data Factory | hexa1056adf

Search factory and documentation

Copy Data tool

Source data store

Specify the source data store for the copy task. You can use an existing data store connection or specify a new data store.

Source type: Azure Blob Storage

Connection: Select... + New connection

Connect via integration runtime: AutoResolveIntegrationRuntime

Authentication type: Account key

Account selection method: From Azure subscription

Azure subscription: Azure subscription 1 (984f097c-963c-4eb6-a20d-839457ae9f08)

Storage account name: hexa1056sourcestorage

Additional connection properties: + New

Test connection: To linked service

Connection successful

Create Cancel

Microsoft Azure | Data Factory | hexa1056adf

Search factory and documentation

Copy Data tool

Source data store

Specify the source data store for the copy task. You can use an existing data store connection or specify a new data store.

Source type: Azure Blob Storage

Connection: hexa1056sourcestorage Edit + New connection

File or folder: hexa1056sourcecontainer/ Browse

Options

Binary copy: ☐

Recursively: ☒

Enable partitions discovery: ☐

Max concurrent connections: 1

Filter by last modified

Start time (UTC): End time (UTC):

Next Cancel

- Create connection with sink storage and mention the path or container destination where the data has to be copy from source storage.

This screenshot shows the 'New connection' dialog in the Microsoft Azure Data Factory interface. The left sidebar indicates the 'Destination' step of the 'Copy Data tool' configuration. The main panel is titled 'Destination data store' and contains the following fields:

- Destination type:** Azure Data Lake Storage Gen2
- Connection:** Select... (with a '+ New connection' link)
- Connect via integration runtime:** AutoResolveIntegrationRuntime
- Authentication type:** Account key
- Account selection method:** From Azure subscription (selected), Enter manually
- Azure subscription:** Azure subscription 1 (984f097c-963c-4eb6-a20d-839457ae9f08)
- Storage account name:** hexa1056sinkstorage
- Test connection:** To linked service (selected), To file path
- Annotations:** + New
- Parameters:** > Parameters

At the bottom right, a green checkmark indicates 'Connection successful', and a 'Test connection' link is available. 'Create' and 'Cancel' buttons are at the bottom.

This screenshot shows the 'Destination' configuration page in the Microsoft Azure Data Factory interface. The left sidebar indicates the 'Destination' step of the 'Copy Data tool' configuration. The main panel is titled 'Destination data store' and contains the following fields:

- Destination type:** Azure Data Lake Storage Gen2
- Connection:** hexa1050sinkstorage (with 'Edit' and '+ New connection' links)
- Folder path:** hexa1050sinkcontainer/ (with a 'Browse' button)
- File name:** (empty text field)
- Copy behavior:** Select... (dropdown menu)
- Max concurrent connections:** (empty text field)
- Block size (MB):** (empty text field)
- Metadata:** + New

At the bottom, there are '< Previous' and 'Next >' buttons, and a 'Cancel' button on the right.

- Give pipeline details, review and start the copy process

Microsoft Azure | Data Factory | hexa1056adf | Search factory and documentation | azuser1056_mmllocal@ihl.onmicrosoft.com

Copy Data tool

Summary
You are running pipeline to copy data from Azure Blob Storage to Azure Data Lake Storage Gen2.

Properties
Task name: CopyActivity1056
Task description: [Edit]

Source
Connection name: hexa1056sourcestorage
Dataset name: SourceDataset_rfq
Column delimiter: ,
Escape character: \
Quote char: '
First row as header: true
Container: hexa1056sourcecontainer

< Previous | Next > | Cancel

Microsoft Azure | Data Factory | hexa1056adf | Search factory and documentation | azuser1056_mmllocal@ihl.onmicrosoft.com

Copy Data tool

Deployment complete

Deployment step	Status
Validating copy runtime environment	✓ Succeeded
> Creating datasets	✓ Succeeded
> Creating pipelines	✓ Succeeded
> Running pipelines	✓ Succeeded

Datasets and pipelines have been created. You can now monitor and edit the copy pipelines or click finish to close Copy Data Tool.

Finish | Edit pipeline | Monitor

Microsoft Azure | Data Factory | hexa1056adf

Search factory and documentation

azuser1056_mm1.local@ihltonmicrosoft.com

» Dashboards
» Runs
» Pipeline runs
» Trigger runs
» Change Data Capture (previ...
» Runtimes & sessions
» Integration runtimes
» Data flow debug
» Notifications
» Alerts & metrics

All pipeline runs > CopyActivity1056 - Activity runs

Rerun Cancel Refresh Update pipeline List Gantt

Pipeline was modified after this run. The current pipeline configuration is shown.

Copy data
Copy_rfq

Activity runs

Pipeline run ID a782b4d0-e8fb-4362-8f9b-6cce866e4878

All status Monitor in Azure Metrics Export to CSV

Showing 1 - 1 items

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties	Destination
Copy_rfq	Succeeded	Copy data	2/26/2024, 11:34:10 PM	13s	AutoResolveIntegrator		hexa1056sinkcontainer. h

- After the process gets finished we can see that the data in the source storage (blob storage) has copied to sink storage (Azure data lake)

Microsoft Azure

Search resources, services, and docs (G+J)

azuser1056_mm1.local@ihltonmicrosoft.com

Home > hexa1056sinkstorage | Containers >

hexa1056sinkcontainer

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: hexa1056sinkcontainer

Search blobs by prefix (case-sensitive) Show deleted objects

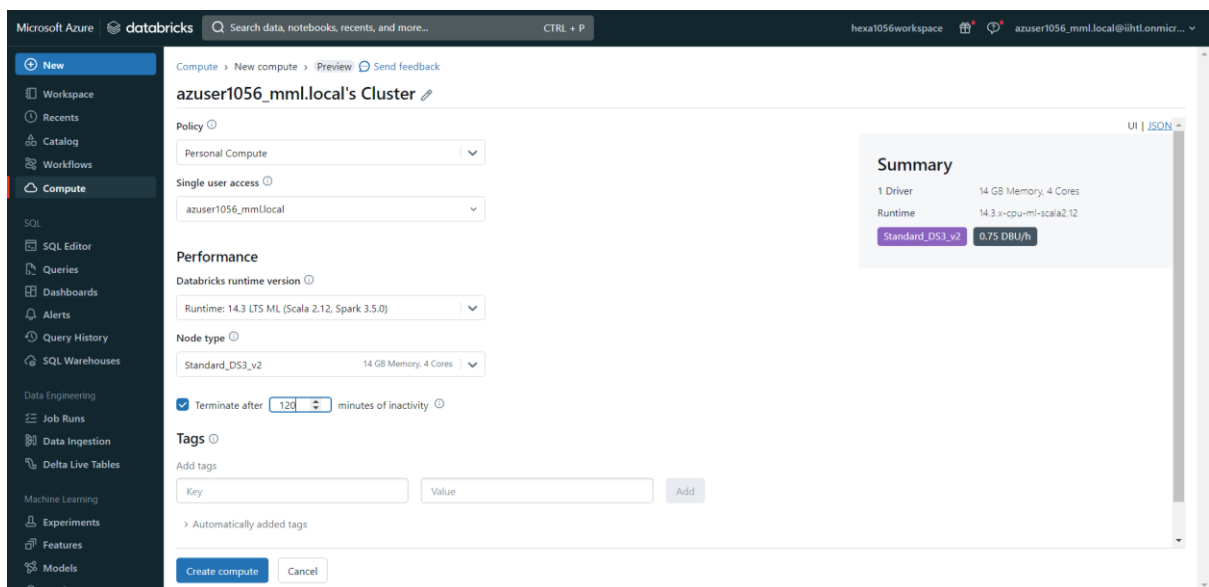
Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
Employee_data.csv	26/02/2024, 23:34:21	Hot (Inferred)		Block blob	9.57 MiB	Available

3. Developing Pyspark Notebook

- Create a new PySpark notebook within the Databricks workspace. Begin writing PySpark code to perform ETL operations, data transformations, and data analytical tasks

4. Create cluster and connecting it to notebook

- The cluster is created with autoscaling is enabled which automatically adjust cluster size to accommodate changes in workload demand, allowing for seamless scalability without manual intervention.



5. Importing Necessary libraries and Creating Spark Session

- Use `SparkSession.builder` to configure and create a `SparkSession`. specify the application name using `.appName()` and configure any additional Spark options using `.config()`. Finally, call `.getOrCreate()` to either create a new `SparkSession`

```
▶ ✓ 39 minutes ago (<1s) Cell 1

# Import the necessary modules
from pyspark.sql import SparkSession
from pyspark.sql.functions import trim, when, col, regexp_replace,sum, count, avg, min, max
from pyspark.sql.window import Window
```

```
▶ ✓ 46 minutes ago (<1s) Cell 2

# Create a SparkSession
spark = SparkSession.builder.appName("DataLakeAnalytics").getOrCreate()
```

6. Extracting Data from Source storage

- Connecting data source (Azure Blob Storage) by mounting it to the Databricks File System (DBFS) to simplify data access
- It helps to retrieve raw data for processing and analysis within the PySpark environment

```
43 minutes ago (11s) Cell 3

# 1) Extracting the data from Data lake
# Mounting the Data Lake with Azure databricks

dbutils.fs.mount(
    source = "wasbs://hexa1050sinkcontainer@hexa1050sinkstorage.blob.core.windows.net",
    mount_point="/mnt/blobStorage",
    extra_configs={"fs.azure.account.key.hexa1050sinkstorage.blob.core.windows.net": "FDVopi9Fy8FwvYxDFNCLDK1Yf/EoFaV0esH9nYNYXjATF7J/6kG+FKP6yvNsuGu/ptbbg1Nm1/MF+ASwMFw=="})

True
```

```
43 minutes ago (<1s) Cell 4

# Listing the File information to get file path

dbutils.fs.ls('/mnt/blobStorage')

[FileInfo(path='dbfs:/mnt/blobStorage/Employee_data.csv', name='Employee_data.csv', size=10034689, modificationTime=1708969782000)]
```

```
43 minutes ago (12s) Cell 5

# Reading the data of Data lake and converting it to spark RDD

path = "dbfs:/mnt/blobStorage/Employee_data.csv"
RDD = spark.read.csv(path, header=True,inferSchema=True)

(2) Spark Jobs

RDD: pyspark.sql.dataframe.DataFrame = [EEID: integer, Full Name: string ... 10 more fields]
```

```
42 minutes ago (1s) Cell 6

RDD.display()

(1) Spark Jobs
```

	EEID	Full Name	Job Title	Department	Gender	Ethnicity	Age
1	98278	Samaira Raj	Human Resources Manager	Human Resources	Male	White	46
2	19840	Lavanya Hayer	Data Analyst	Research and Development	Female	null	55
3	22487	Shayak Raval	Financial Analyst	Finance	Female	White	29
4	17160	Inaaya Bala	Software Engineer	Research and Development	Male	null	65
5	10385	Aarav Garde	Data Analyst	Research and Development	Male	White	56
6	28297	Romil Keer	Customer Service Representative	Customer Service	null	null	19
7	21123	Pranav Chadha	Human Resources Manager	Human Resources	Male	Hispanic	46

10,000 rows | Truncated data | 1.15 seconds runtime

Refreshed 42 minutes ago

7. Transforming the raw data

- Utilize PySpark DataFrame transformations and functions to cleanse, transform, and prepare the data for analysis.
- Implement business logic and data processing steps to transform raw dataset up to mark for data analysis purpose.

Transformations done:-

▪ Removing the Duplicate records

```
▶ Just now (4s) Cell 7

# Removing the Duplicate data

print(RDD.count())
RDD=RDD.distinct()
print(RDD.count())

▶ (5) Spark Jobs
▶ RDD: pyspark.sql.dataframe.DataFrame = [EEID: integer, Full Name: string ... 10 more fields]
100000
99996
```

▪ Handling anonymous data

```
▶ Just now (3s) Cell 8

# Removing the anonymous data

print(RDD.count())
RDD = RDD.na.drop(["any", subset=["EEID"]])
print(RDD.count())

▶ (6) Spark Jobs
▶ RDD: pyspark.sql.dataframe.DataFrame = [EEID: integer, Full Name: string ... 10 more fields]
99996
99984
```

- Removing Extra spaces and filling the null data with proper messages

▶ ✓ 42 minutes ago (<1s) Cell 9

Removing Leading and Trailing spaces from the data

```
RDD = RDD.withColumn("Full Name", trim("Full Name"))
RDD = RDD.withColumn("Job Title", trim("Job Title"))
RDD = RDD.withColumn("Department", trim("Department"))
RDD = RDD.withColumn("Gender", trim("Gender"))
RDD = RDD.withColumn("Ethnicity", trim("Ethnicity"))
RDD = RDD.withColumn("Country", trim("Country"))
```

▶ RDD: pyspark.sql.dataframe.DataFrame = [EEID: integer, Full Name: string ... 10 more fields]

▶ ✓ 42 minutes ago (<1s) Cell 10

Filling the null values with proper message

```
RDD = RDD.na.fill(value="Not Known",subset=["Full Name"])
RDD = RDD.na.fill(value="Not Known",subset=["Job Title"])
RDD = RDD.na.fill(value="Not Known",subset=["Department"])
RDD = RDD.na.fill(value="Prefer Not to say",subset=["Gender"])
RDD = RDD.na.fill(value="Not Known",subset=["Ethnicity"])
RDD = RDD.na.fill(value="Not Known",subset=["Country"])
RDD = RDD.na.fill(value=0,subset=["Bonus %"])
RDD = RDD.withColumn('Hire Date',when(col('Hire Date').isNull(),('No data provided')).otherwise(col('Hire Date')))
RDD = RDD.withColumn('Exit Date',when(col('Exit Date').isNull(),('Currently Working')).otherwise(col('Exit Date')))
```

▶ RDD: pyspark.sql.dataframe.DataFrame = [EEID: integer, Full Name: string ... 10 more fields]

- Handling the numerical columns and renaming USA to US to make dataset consistent

▶ ✓ 41 minutes ago (<1s) Cell 11

```
# Filling the numerical column's null values with proper average values

window_spec = Window.partitionBy()
RDD = RDD.withColumn('Age', when(col('Age').isNull(), avg(col('Age')).over(window_spec)).otherwise(col('Age'))))

average_salaries = RDD.groupBy("Country", "Department").avg("Annual Salary")
RDD = RDD.join(average_salaries, ["Country", "Department"], "left").withColumnRenamed("avg(Annual Salary)", "Average Salary")
RDD = RDD.withColumn('Annual Salary', when(col('Annual Salary').isNull(), col('Average Salary'))
| | | | | .otherwise(col('Annual Salary'))).drop("Average Salary")
```

▶ average_salaries: pyspark.sql.dataframe.DataFrame = [Country: string, Department: string ... 1 more field]
▶ RDD: pyspark.sql.dataframe.DataFrame = [Country: string, Department: string ... 10 more fields]

▶ ✓ 40 minutes ago (1s) Cell 12

```
# Renaming USA as US

RDD=RDD.withColumn('Country',when(RDD.Country=='USA',regexp_replace(RDD.Country, 'USA', 'US')).otherwise(RDD.Country))
print(RDD.select("Country").distinct().collect())
```

▶ (3) Spark Jobs

▶ RDD: pyspark.sql.dataframe.DataFrame = [Country: string, Department: string ... 10 more fields]
[Row(Country='US'), Row(Country='UK'), Row(Country='Canada'), Row(Country='Australia')]

8. Data Analytics

- Data analytics is the collection, transformation, and organisation of data in order to draw conclusions, make predictions, and drive informed decision making.

Analytics Performed

- Grouping the data based on country and department (count_df)

▶ ▼

✓ 40 minutes ago (1s)

Cell 14

```
# Employee count based on country and department

count_df = RDD.groupBy("Country", "Department").agg(
    count("EEID").alias("Total_employees"),
    sum(when(RDD["Gender"] == "Male", 1).otherwise(0)).alias("Male"),
    sum(when(RDD["Gender"] == "Female", 1).otherwise(0)).alias("Female"),
    sum(when(RDD["Gender"].isin("Male", "Female"), 0).otherwise(1)).alias("Prefer_not_to_say"),
    sum(when(RDD["Ethnicity"] == "Black", 1).otherwise(0)).alias("Black"),
    sum(when(RDD["Ethnicity"] == "White", 1).otherwise(0)).alias("White"),
    sum(when(RDD["Ethnicity"] == "Asian", 1).otherwise(0)).alias("Asian"),
    sum(when(RDD["Ethnicity"].isin("Black", "White", "Asian"), 0).otherwise(1)).alias("other_race"),
    sum(when(RDD["Age"] < 40, 1).otherwise(0)).alias("age_below_40"),
    sum(when(RDD["Age"] > 40, 1).otherwise(0)).alias("age_above_40")
).orderBy("Country", "Department")
count_df.display()
```

▶ (3) Spark Jobs

▶ count_df: pyspark.sql.dataframe.DataFrame = [Country: string, Department: string ... 10 more fields]

Table ▼ +

New result table: OFF ▼

	Country	Department	Total_employees	Male	Female	Prefer_not_to_say	Black	White	Asi
1	Australia	Customer Service	2387	792	791	804	426	511	454
2	Australia	Engineering	2499	878	850	771	471	507	498
3	Australia	Finance	2418	804	836	778	468	499	500
4	Australia	Human Resources	2541	861	851	829	481	528	491
5	Australia	IT	2484	817	864	803	520	500	464
6	Australia	Legal	2480	866	826	788	473	488	498
7	Australia	Marketing	2552	851	905	796	464	530	488

↓ 40 rows | 1.39 seconds runtime

Refreshed 40 minutes ago

- Sorting the data by considering only working employees (working_employees_df)

33 minutes ago (2s) Cell 16

```
# Fetching records of working employees

working_employees_df = RDD.filter(col("Exit Date") == "Currently Working")
working_employees_df = working_employees_df.drop("Exit Date")

working_employees_df.display()
```

(5) Spark Jobs

working_employees_df: pyspark.sql.dataframe.DataFrame = [Country: string, Department: string ... 9 more fields]

Table + New result table: OFF

	Country	Department	EEID	Full Name	Job Title	Gender	Ethnicity
1	Australia	Sales	45876	Jayant Devan	Sales Representative	Female	Black
2	Canada	IT	69030	Abram Mani	Software Engineer	Male	Hispanic
3	UK	Customer Service	39341	Kismat Keer	Customer Service Representative	Female	Asian
4	Canada	Engineering	86184	Stuvan Dutt	Software Engineer	Female	Black
5	US	Operations	78333	Ritvik Lanka	Customer Service Representative	Prefer Not to say	Not Known
6	UK	Engineering	75249	Arnav Mand	Project Manager	Male	Not Known
7	Australia	Marketing	47025	Samar Chowdhury	Marketing Specialist	Male	Not Known

10,000 rows | Truncated data | 1.58 seconds runtime Refreshed 33 minutes ago

- Sorting the data by considering only those employees who left the company (non_working_employees_df)

27 minutes ago (1s) Cell 18

```
# Fetching records of non-working employees

non_working_employees_df = RDD.filter(col("Exit Date") != "Currently Working")

non_working_employees_df.display()
```

(5) Spark Jobs

non_working_employees_df: pyspark.sql.dataframe.DataFrame = [Country: string, Department: string ... 10 more fields]

Table + New result table: OFF

	Country	Department	EEID	Full Name	Job Title	Gender	Ethnicity
1	Canada	Marketing	46400	Riya Grover	Data Analyst	Female	Asian
2	UK	Engineering	61085	Ira Wable	Project Manager	Female	Asian
3	US	IT	92676	Parinaaz Karpe	Software Engineer	Female	Asian
4	Canada	Research and Development	38396	Taran Butala	Software Engineer	Female	White
5	Canada	Sales	22677	Samiha Vasa	Sales Representative	Female	Not Known
6	US	Customer Service	94951	Kabir Dey	Customer Service Representative	Prefer Not to say	Not Known
7	UK	Sales	30669	Tarini Brahmabhatt	Sales Representative	Male	White

10,000 rows | Truncated data | 1.36 seconds runtime Refreshed 27 minutes ago

- Making the statistical data of Annual salary and Bonus percentage in each country based on department (salaries_df)

12:01 AM (1s)

Cell 20

Country and department wise salaries

salaries_df = RDD.groupBy("Country", "Department").agg(
 avg(RDD["Annual Salary"]).alias("Average_Salary"),
 min(RDD["Annual Salary"]).alias("Min_Salary"),
 max(RDD["Annual Salary"]).alias("Max_Salary"),
 avg(RDD["Bonus %"]).alias("Average_Bonus"),
 min(RDD["Bonus %"]).alias("Min_Bonus"),
 max(RDD["Bonus %"]).alias("Max_Bonus")
)

salaries_df.display()

(5) Spark Jobs

salaries_df: pyspark.sql.dataframe.DataFrame = [Country: string, Department: string ... 6 more fields]

Table

New result table: OFF

	Country	Department	Average_Salary	Min_Salary	Max_Salary	Average_Bonus	Min_Bonus	Max_Bonus
1	US	Marketing	79991.36395141699	40086.07	119939.58	4.877534412955467	0	10
2	US	Operations	79234.15896837942	40033.76	119974.62	5.003936758893281	0	10
3	US	Sales	79589.61770023788	40010.3	119984.9	5.02660190325139	0.01	10
4	UK	Legal	81114.15970008212	40056.76	119986.21	5.086400986031222	0	10
5	UK	IT	80053.21730563555	40055.65	119992.87	4.991616618675444	0	9.99
6	Canada	Engineering	79417.14880513234	40023.06	119980.68	5.043031275060145	0.01	10
7	US	IT	80151.04563575322	40030.79	119971.89	5.014553084347462	0	10

40 rows | 1.15 seconds runtime

Refreshed 20 minutes ago

9. Loading and organizing the analytical data to the azure data lake

- By using the mount point load and organize the data into the data lake
- Use the appropriate folder path in the container for better organizing of analytical data and also use the appropriate names for the files

```
12:07 AM (6s) Cell 22

# writing the analytical data to the data lake

count_df = count_df.toPandas()
count_df.to_csv("/dbfs/mnt/blobStorage/output/total_employee.csv", index = False)

working_employees_df = working_employees_df.toPandas()
working_employees_df.to_csv("/dbfs/mnt/blobStorage/output/working_employee.csv", index = False)

non_working_employees_df = non_working_employees_df.toPandas()
non_working_employees_df.to_csv("/dbfs/mnt/blobStorage/output/non_working_employee.csv", index = False)

salaries_df = salaries_df.toPandas()
salaries_df.to_csv("/dbfs/mnt/blobStorage/output/salaries.csv", index = False)

(18) Spark Jobs
```

- We can see the analytical data files has been generated and organized in the appropriate output path in the azure data lake

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/> non_working_employee.csv	27/02/2024, 19:21:58	Hot (Inferred)		Block blob	5.42 MiB	Available
<input type="checkbox"/> salaries.csv	27/02/2024, 19:22:01	Hot (Inferred)		Block blob	3.25 KiB	Available
<input type="checkbox"/> total_employee.csv	27/02/2024, 19:21:52	Hot (Inferred)		Block blob	2.48 KiB	Available
<input type="checkbox"/> working_employee.csv	27/02/2024, 19:21:55	Hot (Inferred)		Block blob	4.88 MiB	Available

10. Unmounting the azure data lake

▶ ✓ 12:10 AM (11s)

Unmounting the Data lake

```
dbutils.fs.unmount("/mnt/blobStorage")
```

/mnt/blobStorage has been unmounted.

True

1. Grouped data based on country and department

	A	B	C	D	E	F	G	H	I	J	K	L
1	Country	Department	Total_emp	Male	Female	Prefer_not	Black	White	Asian	other_race	age_below	age_above
2	Australia	Customer	2387	792	791	804	426	511	454	996	980	1356
3	Australia	Engineering	2499	878	850	771	471	507	498	1023	1019	1427
4	Australia	Finance	2418	804	836	778	468	499	500	951	1008	1377
5	Australia	Human Resources	2541	861	851	829	481	528	497	1035	1044	1436
6	Australia	IT	2484	817	864	803	520	500	466	998	990	1439
7	Australia	Legal	2480	866	826	788	473	488	499	1020	1007	1435
8	Australia	Marketing	2552	851	905	796	464	530	488	1070	1051	1451
9	Australia	Operations	2462	761	860	841	495	490	500	977	1015	1406
10	Australia	Research & Development	2488	820	833	835	477	492	511	1008	1030	1406
11	Australia	Sales	2610	899	888	823	519	487	520	1084	1107	1456
12	Canada	Customer	2552	856	843	853	509	510	504	1029	1082	1418
13	Canada	Engineering	2494	846	812	836	498	504	532	960	1050	1398
14	Canada	Finance	2557	850	826	881	500	509	519	1029	1075	1439
15	Canada	Human Resources	2509	843	862	804	490	503	510	1006	1037	1414
16	Canada	IT	2566	845	874	847	507	473	552	1034	1120	1395
17	Canada	Legal	2597	858	904	835	526	537	513	1021	1085	1478
18	Canada	Marketing	2512	826	869	817	511	484	557	960	1023	1440
19	Canada	Operations	2469	822	851	796	505	509	497	958	1071	1359
20	Canada	Research & Development	2579	794	875	910	537	541	518	983	1078	1454
21	Canada	Sales	2572	830	840	902	515	507	542	1008	1083	1443
22	UK	Customer	2497	859	842	796	492	493	502	1010	1006	1437
23	UK	Engineering	2445	815	849	781	471	496	479	999	1030	1361
24	UK	Finance	2508	877	817	814	484	502	526	996	1041	1411
25	UK	Human Resources	2535	818	847	870	506	505	510	1014	1042	1462
26	UK	IT	2431	807	806	818	493	435	504	999	993	1395
27	UK	Legal	2434	802	847	785	486	499	478	971	1041	1352
	<	>	total_employee	(+)								

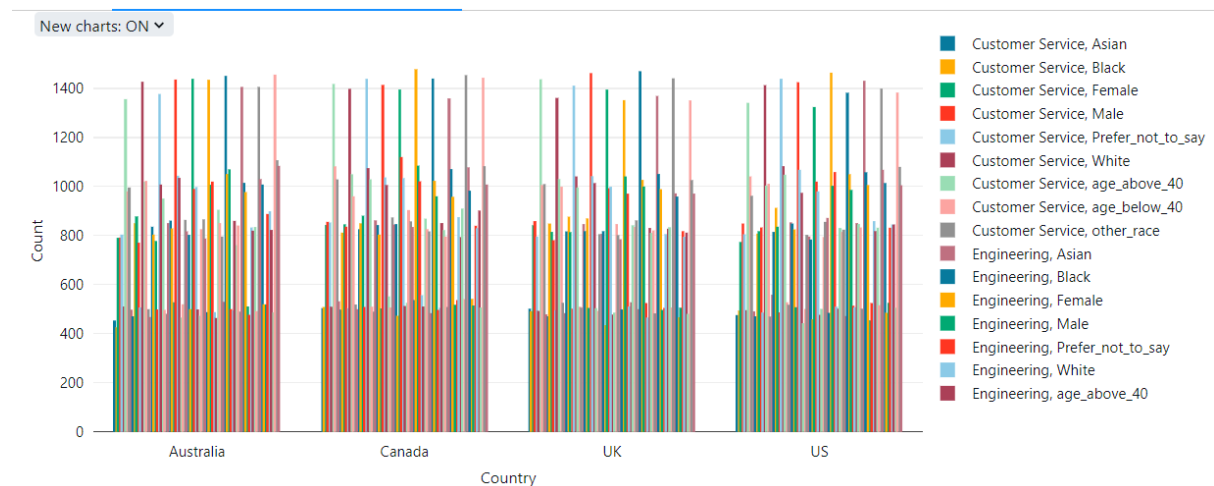
3. Non-Working employees data

	A	B	C	D	E	F	G	H	I	J	K	L
1	Country	Department	EEID	Full Name	Job Title	Gender	Ethnicity	Age	Hire Date	Annual Sal	Bonus %	Exit Date
2	Canada	Marketing	46400	Riya Grove	Data Analyst	Female	Asian	25	#####	75499.29	3.83	#####
3	UK	Engineering	61085	Ira Wable	Project Manager	Female	Asian	18	#####	110198.8	2.09	#####
4	US	IT	92676	Parinaaz K	Software Engineer	Female	Asian	44	#####	41026.82	5.31	#####
5	Canada	Research & Development	38396	Taran Butcher	Software Engineer	Female	White	68	#####	105485.7	8.69	#####
6	Canada	Sales	22677	Samiha Vaidya	Sales Representative	Female	Not Known	70	#####	104495.8	5.83	#####
7	US	Customer Support	94951	Kabir Dey	Customer Support	Prefer Not To Say	Not Known	40	#####	49184.54	8.37	#####
8	UK	Sales	30669	Tarini Brahmachari	Sales Representative	Male	White	64	#####	64526.56	9.09	#####
9	UK	Sales	44291	Aradhya Choudhary	Sales Representative	Prefer Not To Say	Not Known	29	#####	71443.58	1.7	#####
10	Canada	Research & Development	20783	Rasha Aurora	Data Analyst	Female	White	40	#####	118531.4	2.32	#####
11	Canada	Legal	38418	Ryan Sura	Lawyer	Prefer Not To Say	Not Known	18	#####	46707.41	4.8	#####
12	US	Operations	45764	Adira Sahn	Customer Support	Male	Not Known	26	#####	78001.94	2.36	#####
13	Australia	IT	86860	Navya Shekar	Data Analyst	Male	Hispanic	49	#####	71266.08	4.21	#####
14	US	Legal	45864	Farhan Baloch	Lawyer	Male	White	22	#####	116168	5.35	#####
15	UK	Finance	31359	Anvi Ahluwalia	Accountant	Prefer Not To Say	Asian	25	#####	101659	9.25	#####
16	UK	Operations	27549	Emir Tailor	Operations	Prefer Not To Say	Black	64	#####	68657.94	5.17	#####
17	Australia	Human Resources	80691	Indrans Lal	Human Resources	Male	White	66	#####	103161.8	0.66	#####
18	Canada	Operations	88868	Biju Sani	Operations	Male	Hispanic	70	#####	59208.66	7.47	#####
19	UK	Finance	46756	Dhruv Chaudhary	Financial Analyst	Female	Not Known	32	#####	82979.87	1.98	#####
20	UK	Customer Support	19905	Miraan Karim	Customer Support	Female	Hispanic	50	#####	82354.25	1.93	#####
21	UK	Finance	81298	Jhanvi Edwar	Accountant	Male	Hispanic	70	#####	101083.8	0.45	#####
22	Canada	Legal	98088	Ehsaan Raza	Lawyer	Male	Hispanic	18	#####	86847.62	3.76	#####
23	US	IT	99341	Navya Reddy	Data Analyst	Female	Black	18	#####	76176.64	7.18	#####
24	Australia	Operations	83584	Parinaaz K	Customer Support	Female	Hispanic	28	#####	73740.33	4.41	#####
25	Australia	Marketing	41923	Fateh Yogi	Marketing	Prefer Not To Say	White	60	#####	77113.86	0.31	#####
26	Canada	Research & Development	94118	Tushar Bala	Data Analyst	Male	Not Known	62	#####	41376.64	2.14	#####
27	UK	Human Resources	45580	Prisha Sarin	Human Resources	Male	Hispanic	42	#####	62170.93	9.64	#####
	<	>	non_working_employee (+)									

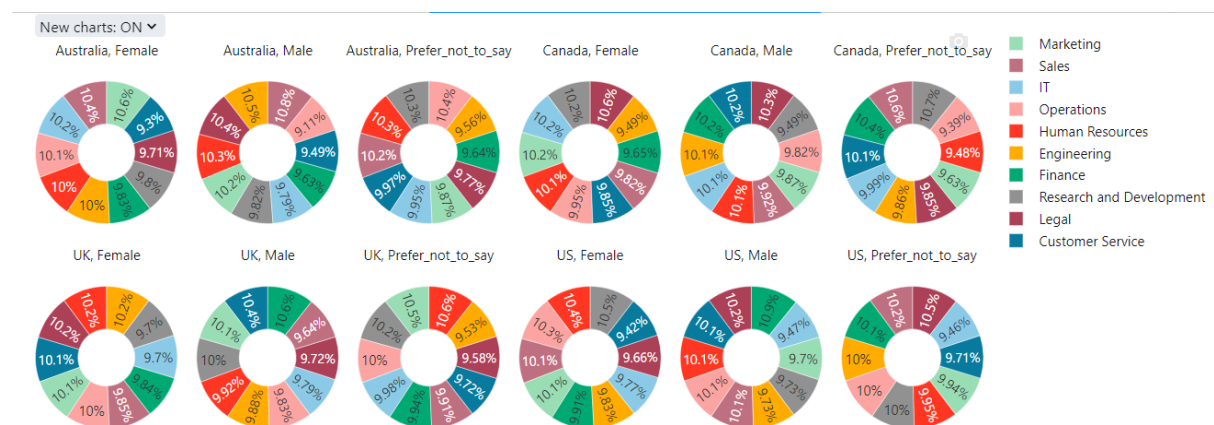
Data visualization on analytical data (truncated data is used)

1. Count_df:

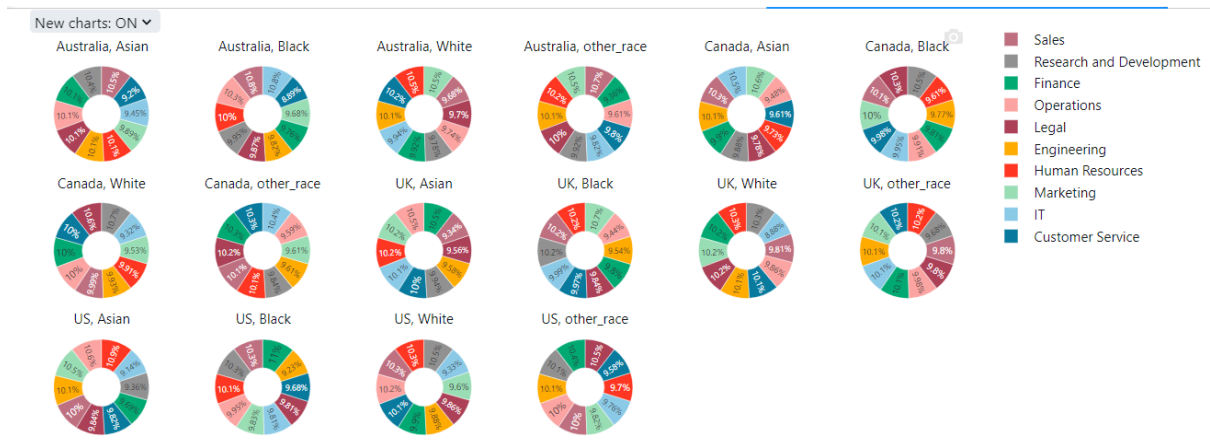
- Bar graph of count_df with country at x-axis and count at y-axis



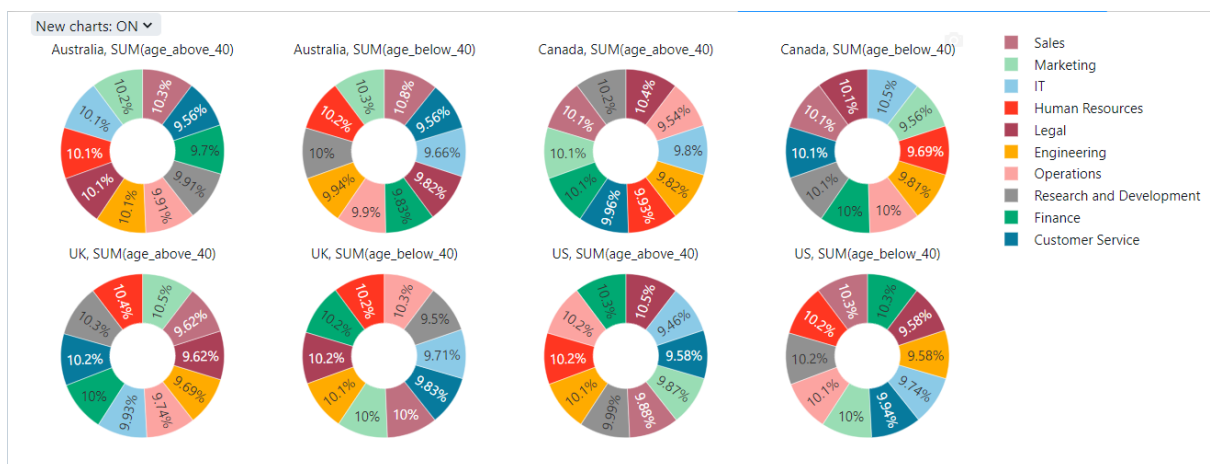
- Pie chart of Departmental Statistical data based on gender



- Pie chart of Departmental Statistical data based on ethnicity

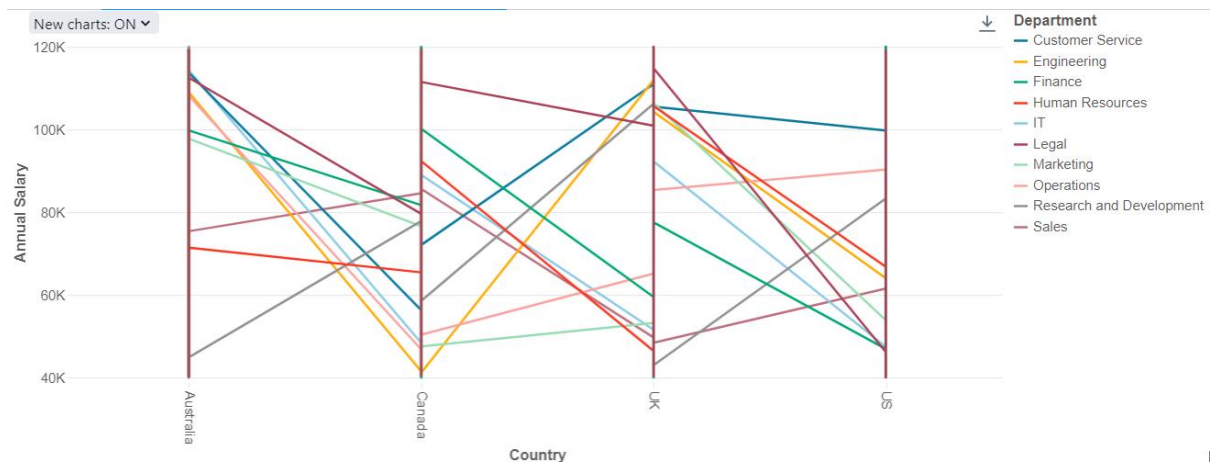


- Pie chart of Departmental Statistical data based on age

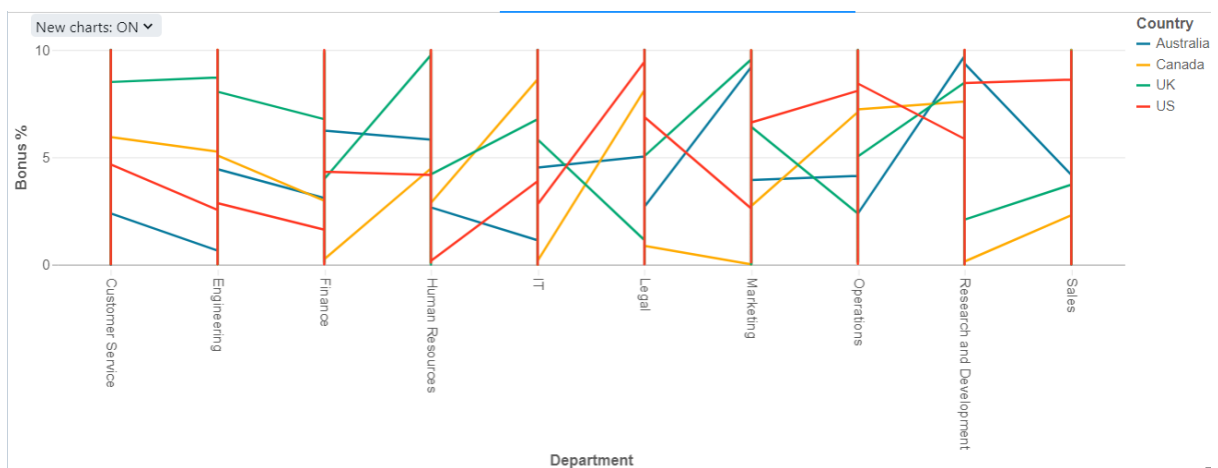


2. working_employee_df

- Representation of Departmental wise Annual Salary for each country

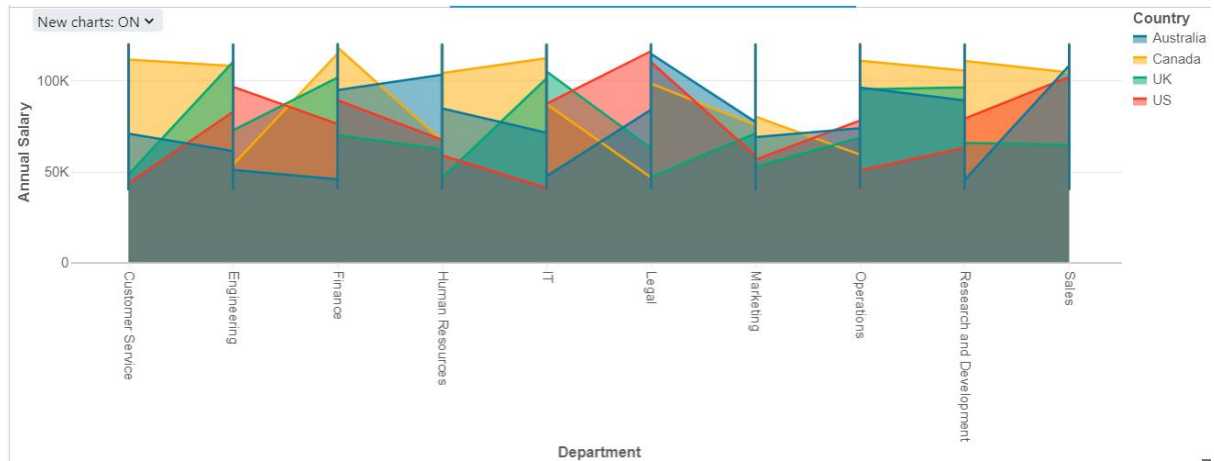


- Representation of Departmental wise Bonus percentage for each country

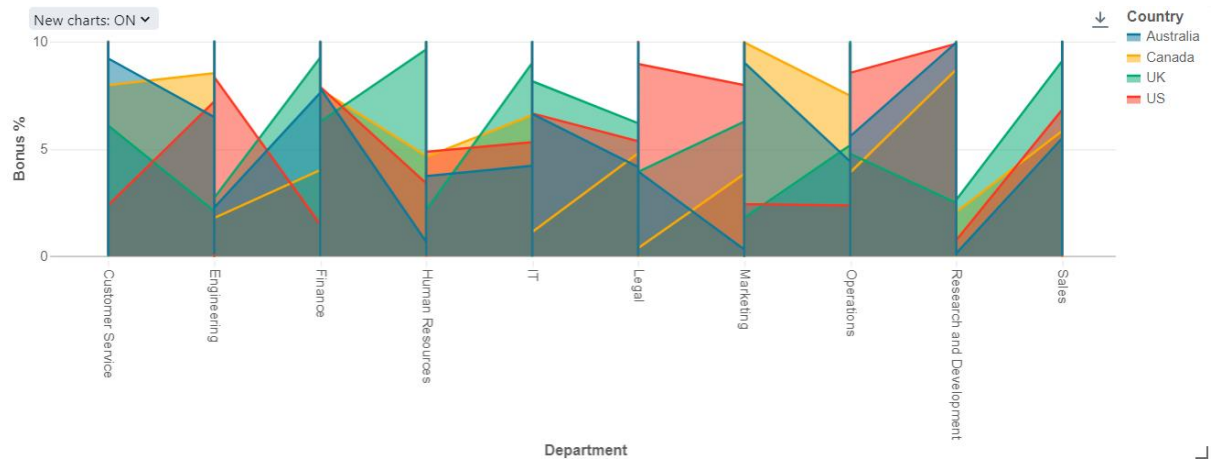


3. non_working_employee_df

- Representation of Departmental wise Annual Salary for each country

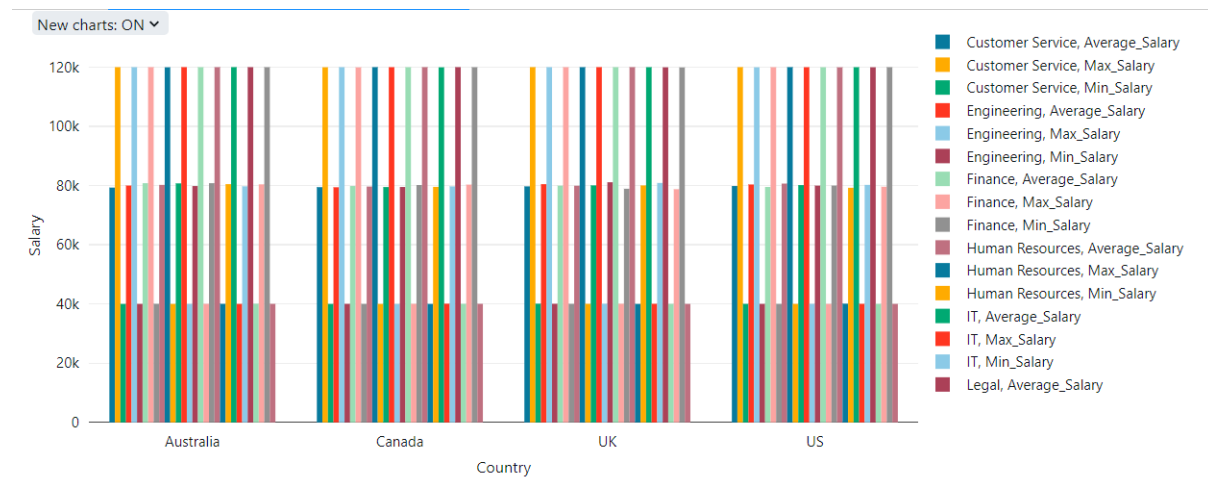


- Representation of Departmental wise Bonus percentage for each country

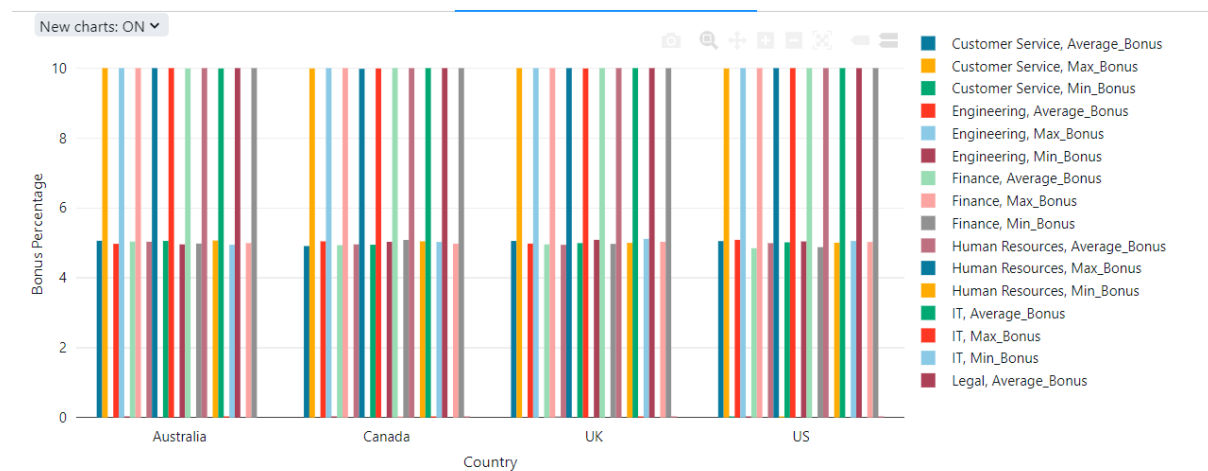


4. salaries_df

- Representation of Department wise Annual Salary for each country



- Representation of Department wise Bonus percentage for each country



Conclusion

In conclusion, this project showcases a cohesive data processing pipeline leveraging Azure Data Factory and Azure Databricks, seamlessly moving data from Azure Blob Storage to Azure Data Lake Storage Gen2 then processing the data by performing transformation and analytical operations through azure databricks, then loading and organizing the data back to Azure data lake storage.