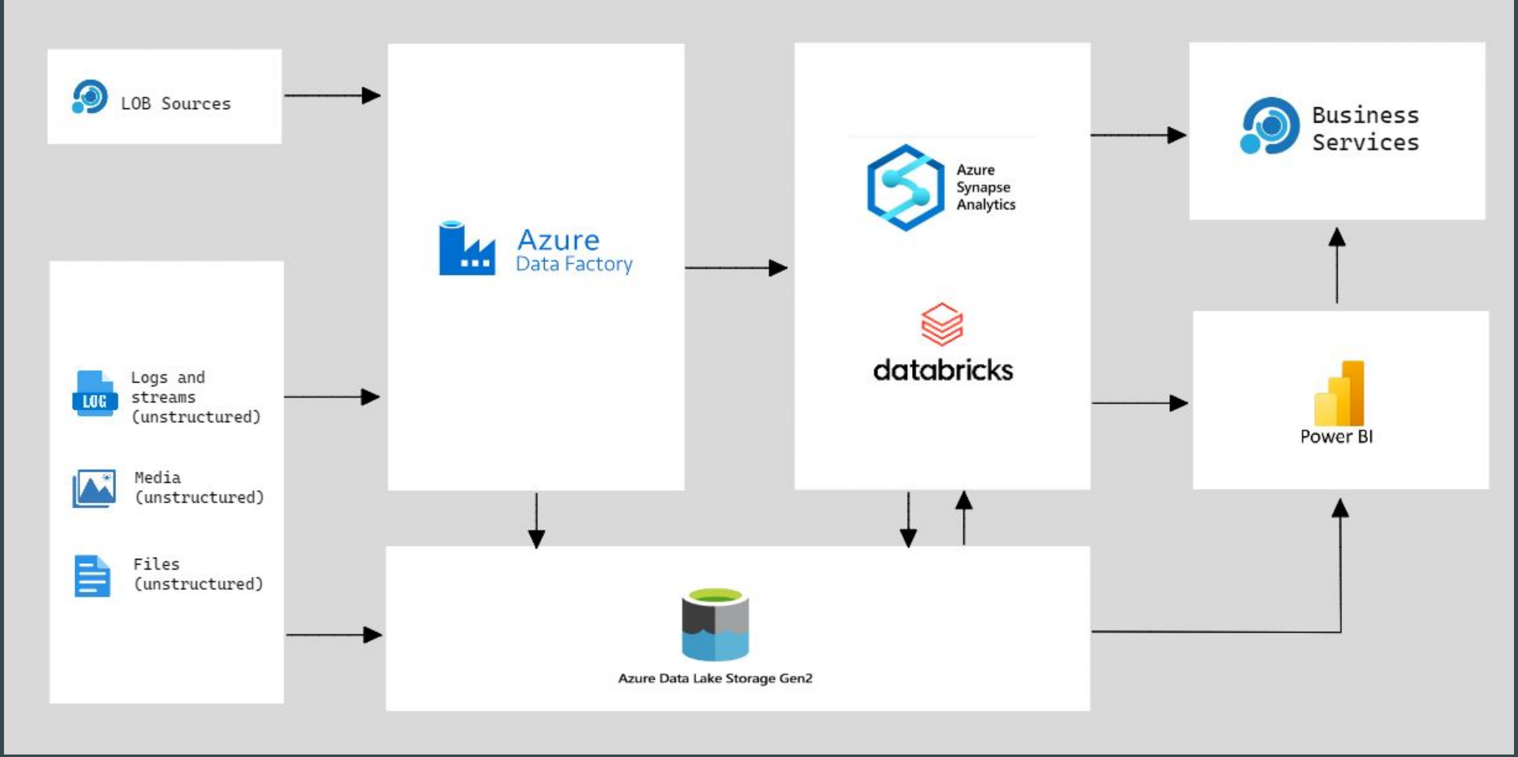
Retail Sales Analysis

Group 06

Roshan Rawat, Gangothri Gadige, Chandan Kumar,
Narendra Reddy, Faryar Memon

Modern Data Warehouse Architecture



Tasks

Source	Azure SQL	● Collecting the data from the client source
ETL	Azure Data Factory	● Using **ADF** to ingest data from client source **Azure SQL Database** to **Azure Data Lake Storage**
Analysis	Azure Synapse Analytics databricks APACHE Spark	● Integrating ADLS with **Synapse** using **Linked Services** & mounting ADLS to **Azure Databricks**
Visualization	Power BI	● Integrating the analysis results with **Power BI** for impactful visualizations and reporting

Managed by DE Expert Venkat Sir✨

Our Sub-Teams:

IAM Team

Managing Azure AD resources and access controls, configuring authentication methods and monitoring access activities

- Faryar
- Roshan

Data Integration Team

Extracting data from various sources, transforming and loading data into data warehouse; maintaining ETL workflows and pipelines

- Gangothri
- Roshan

Data Analytics & BI Team

Perform big data analysis and create dashboards to communicate findings

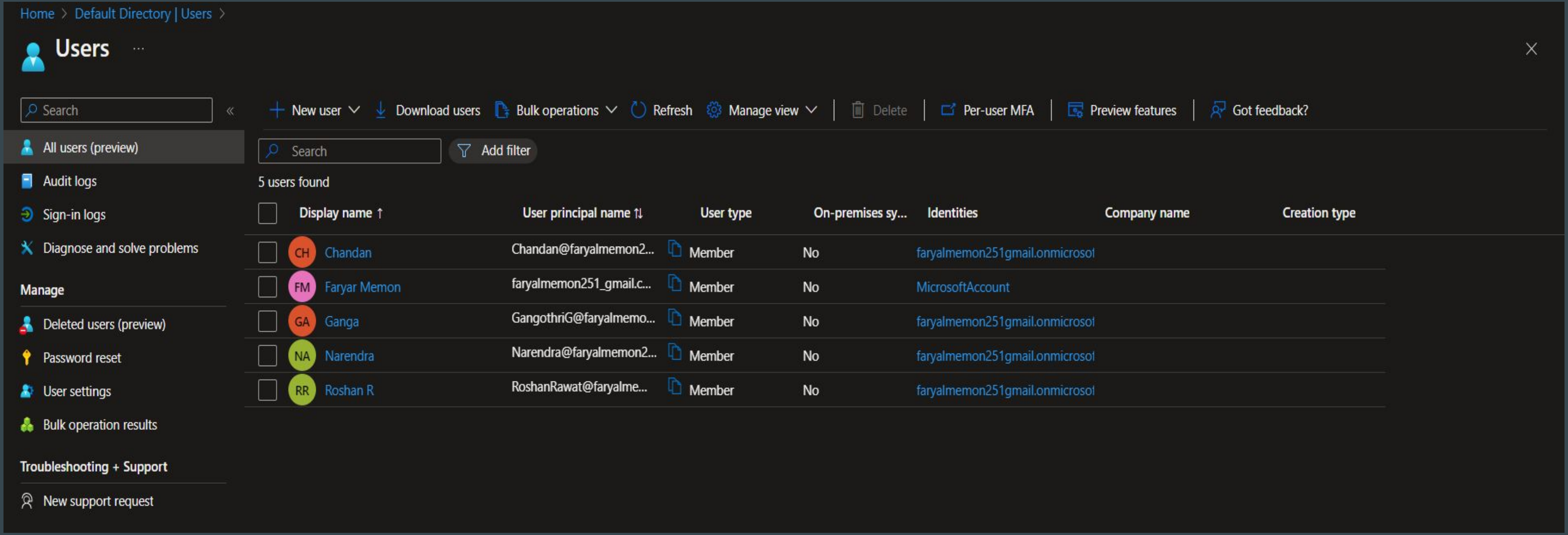
- Chandan
- Narendra
- Roshan

Why add multiple users to a single Resource Group?

- Centralized Access Management
- Seamless Collaboration and Resource Sharing
- Improved Security
- Cost Optimization

This simplifies administration, enhances collaboration, improves security, and optimizes costs

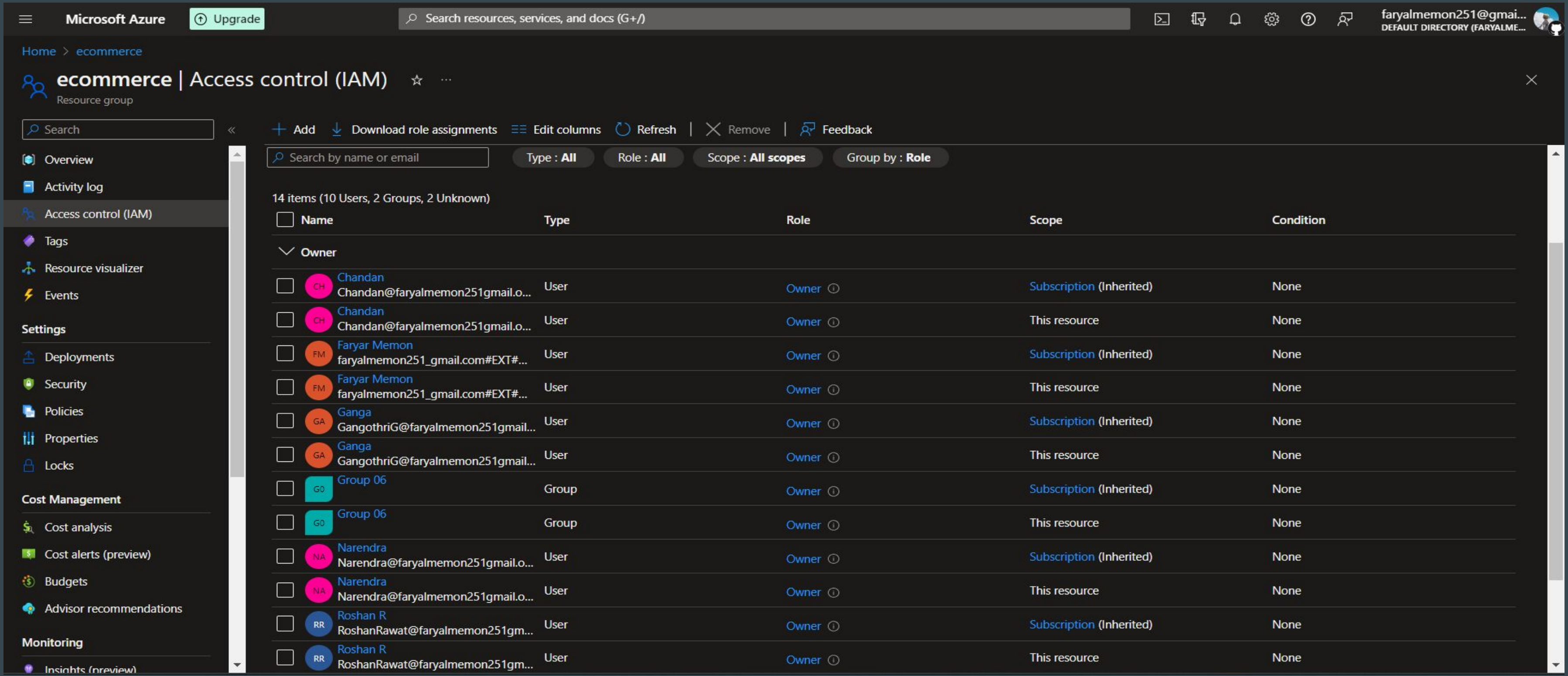
Creating Users Under One Azure Active Directory



5 users found

□	Display name ↑	User principal name ⇅	User type	On-premises sy...	Identities	Company name	Creation type
□	CH Chandan	Chandan@faryalmemon2...	Member	No	faryalmemon251gmail.onmicrosol		
□	FM Faryar Memon	faryalmemon251_gmail.c...	Member	No	MicrosoftAccount		
□	GA Ganga	GangothriG@faryalmemo...	Member	No	faryalmemon251gmail.onmicrosol		
□	NA Narendra	Narendra@faryalmemon2...	Member	No	faryalmemon251gmail.onmicrosol		
□	RR Roshan R	RoshanRawat@faryalme...	Member	No	faryalmemon251gmail.onmicrosol		

Providing Privileges Resource Group and Subscriptions



ecommerce | Access control (IAM)

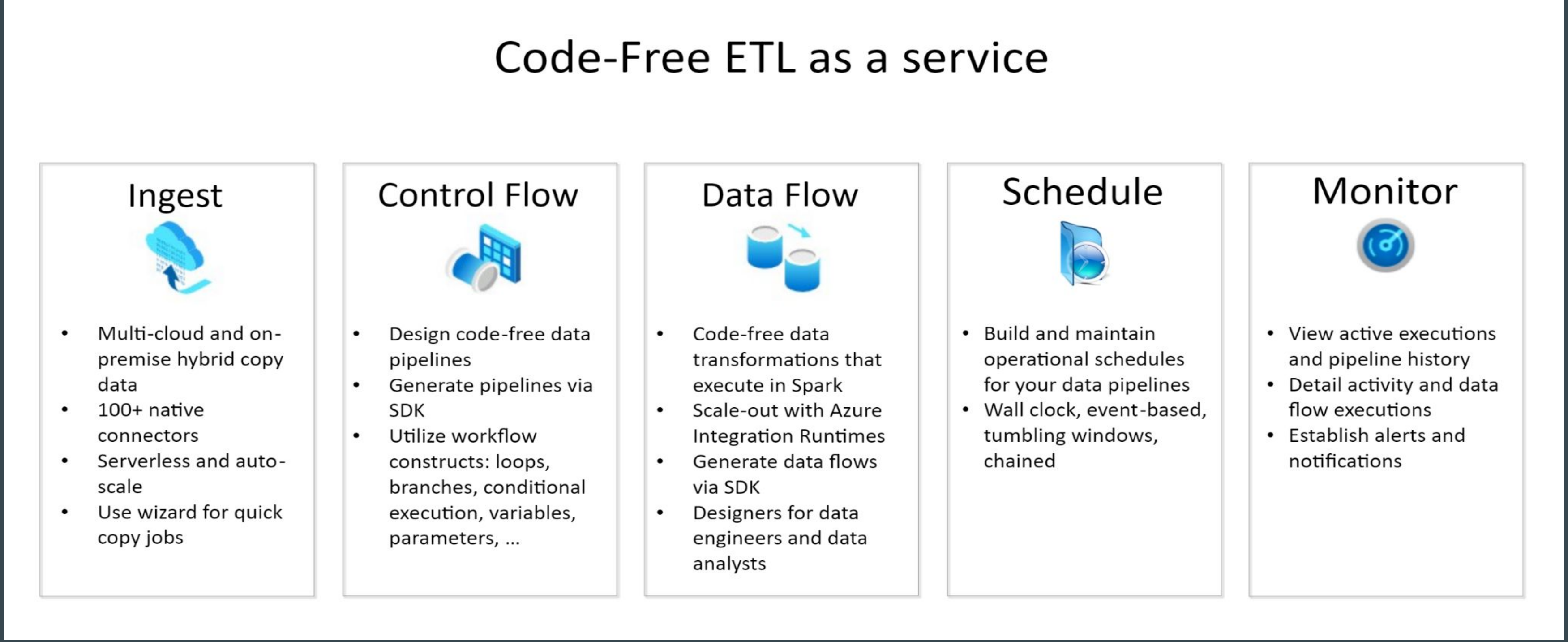
Resource group

14 items (10 Users, 2 Groups, 2 Unknown)

□ Name	Type	Role	Scope	Condition
Owner				
□ CH Chandan Chandan@faryalmemon251gmail.o...	User	Owner ⓘ	Subscription (Inherited)	None
□ CH Chandan Chandan@faryalmemon251gmail.o...	User	Owner ⓘ	This resource	None
□ FM Faryar Memon faryalmemon251_gmail.com#EXT#...	User	Owner ⓘ	Subscription (Inherited)	None
□ FM Faryar Memon faryalmemon251_gmail.com#EXT#...	User	Owner ⓘ	This resource	None
□ GA Ganga GangothriG@faryalmemon251gmail...	User	Owner ⓘ	Subscription (Inherited)	None
□ GA Ganga GangothriG@faryalmemon251gmail...	User	Owner ⓘ	This resource	None
□ G0 Group 06	Group	Owner ⓘ	Subscription (Inherited)	None
□ G0 Group 06	Group	Owner ⓘ	This resource	None
□ NA Narendra Narendra@faryalmemon251gmail.o...	User	Owner ⓘ	Subscription (Inherited)	None
□ NA Narendra Narendra@faryalmemon251gmail.o...	User	Owner ⓘ	This resource	None
□ RR Roshan R RoshanRawat@faryalmemon251gm...	User	Owner ⓘ	Subscription (Inherited)	None
□ RR Roshan R RoshanRawat@faryalmemon251gm...	User	Owner ⓘ	This resource	None

Azure Data Factory

Azure Data Factory



Code-Free ETL as a service

Ingest

- Multi-cloud and on-premise hybrid copy data
- 100+ native connectors
- Serverless and auto-scale
- Use wizard for quick copy jobs

Control Flow

- Design code-free data pipelines
- Generate pipelines via SDK
- Utilize workflow constructs: loops, branches, conditional execution, variables, parameters, …

Data Flow

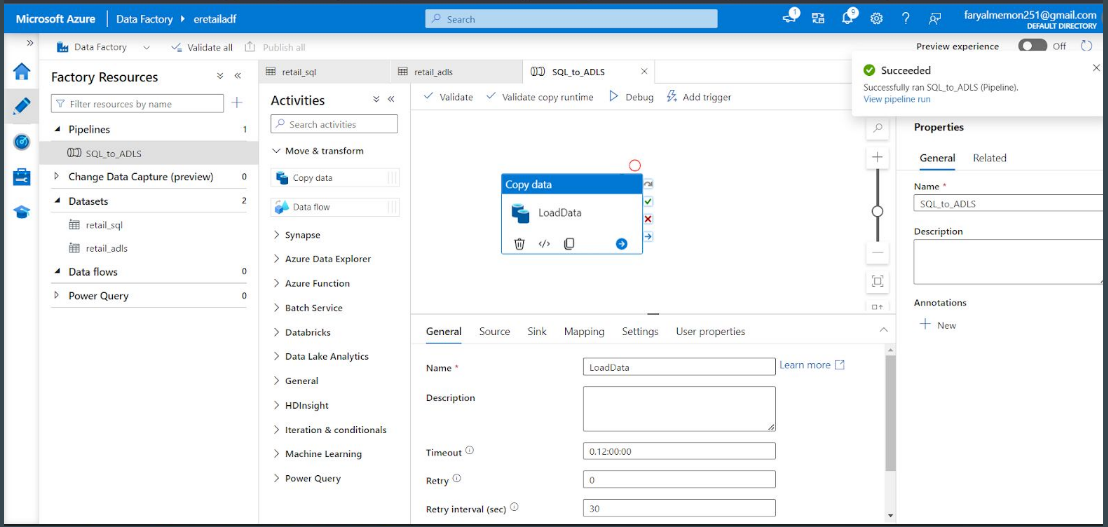
- Code-free data transformations that execute in Spark
- Scale-out with Azure Integration Runtimes
- Generate data flows via SDK
- Designers for data engineers and data analysts

Schedule

- Build and maintain operational schedules for your data pipelines
- Wall clock, event-based, tumbling windows, chained

Monitor

- View active executions and pipeline history
- Detail activity and data flow executions
- Establish alerts and notifications



Succeeded

Successfully ran SQL_to_ADLS (Pipeline).

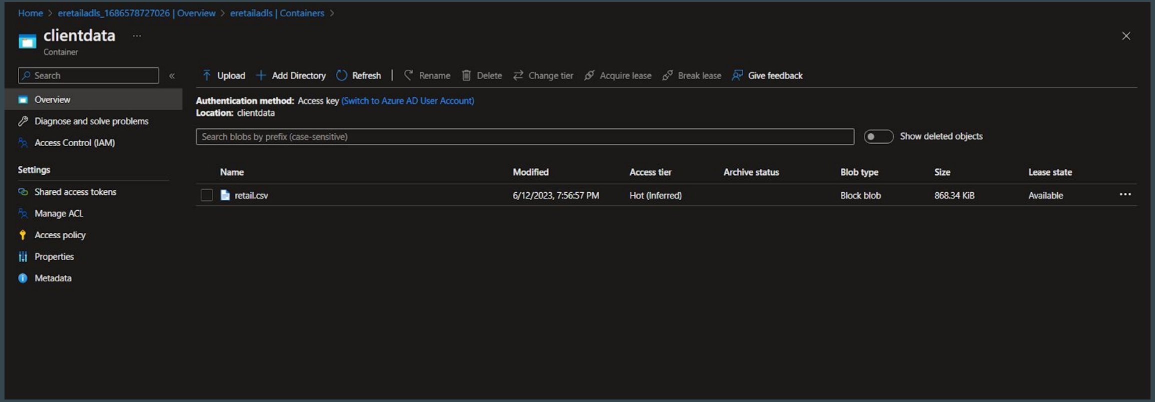
View pipeline run

Name *	LoadData
Description	
Timeout	0.12:00:00
Retry	0
Retry interval (sec)	30

Name *	SQL_to_ADLS
Description	

←–– ADF Pipeline “SQL to ADLS” in action

ADLS Storage –—>

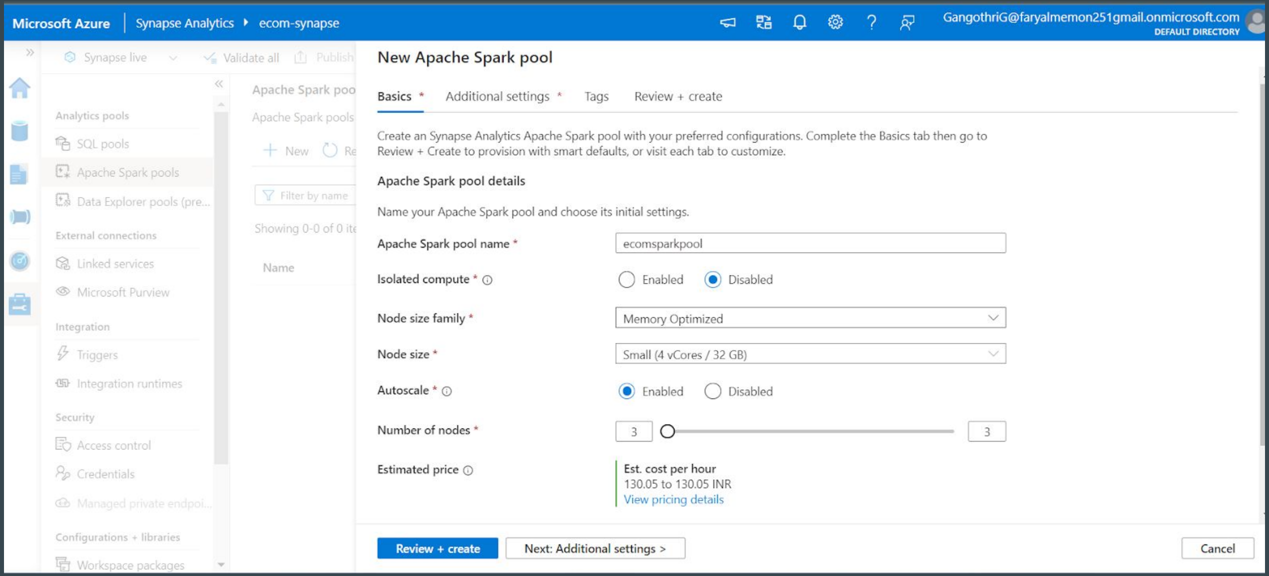


Authentication method: Access key (Switch to Azure AD User Account)
Location: clientdata

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
retail.csv	6/12/2023, 7:56:57 PM	Hot (Inferred)		Block blob	868.34 KiB	Available

Azure Synapse

Synapse Workspace is the overall container that holds all the resources and settings related to Azure Synapse, while **Synapse Studio** is the web-based interface within the workspace that provides tools and features for development, management, and collaboration on Synapse components.



New Apache Spark pool

Create an Synapse Analytics Apache Spark pool with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize.

Apache Spark pool details

Name your Apache Spark pool and choose its initial settings.

Apache Spark pool name * ecomsparkpool

Isolated compute * ○ Enabled ● Disabled

Node size family * Memory Optimized

Node size * Small (4 vCores / 32 GB)

Autoscale * ● Enabled ○ Disabled

Number of nodes * 3 3

Estimated price: Est. cost per hour 130.05 to 130.05 INR View pricing details

Analysis

```
# Loading data:
df=spark.read.load('abfss://source@retailsales123.dfs.core.windows.net/retail.csv
', format='csv'
, header=True,inferSchema=True
)
```

```
# Pre-processing:
retail_df = df.withColumn('order date', to_date(col('Order Date'), 'MM/dd/yyyy'))
            .withColumn('ship date', to_date(col('Ship Date'), 'MM/dd/yyyy'))
```

User Stories

User Story: Monthly Sales Report

```
monthly_sales = retail_df.groupBy([year("Order Date").alias("Years"), month("Order Date").alias("Month")]).agg(sum("Total Cost").alias("Sales")).orderBy("Years","Month")
```

```
## Output:
+-----+-----+---------------------+
|Years|Month| Sales                |
+-----+-----+---------------------+
| 2010|    1| 5.611750021999984E7|
| 2010|    2| 6.14970372599998E7|
| 2010|    3| 3.256335147999997E7|
| 2010|    4| 5.00899801000001E7|
| 2010|    5| 6.25123155799999E7|
| 2010|    6| 5.51885450600001E7|
| 2010|    7| 2.882829051999996E7|
| 2010|    8| 5.16378943899999E7|
| 2010|    9| 4.10263253E7|
| 2010|   10| 4.209123058E7|
| 2010|   11| 3.88184109300001E7|
| 2010|   12| 5.583074315E7|
| 2011|    1| 3.53104419500001E7|
| 2011|    2| 4.812870418000015E7|
| 2011|    3| 4.93220006499999E7|
| 2011|    4| 3.025091864000004E7|
| 2011|    5| 5.09974048800001E7|
| 2011|    6| 4.028547157E7|
| 2011|    7| 6.077437659999994E7|
| 2011|    8| 5.406799758999996E7|
+-----+-----+---------------------+
```

User Story: Quarterly Sales Report

```
quarterly_sales_total_revenue = retail_df.groupBy([year("Order Date").alias("Year"), quarter("Order Date").alias("Quarter")]).agg(sum("Total Revenue")).orderBy("Year","Quarter")
```

```
## Output:
+----+--------+--------------------+
|Year|Quarter| sum(Total Revenue) |
+----+--------+--------------------+
|2010|  1|2.156864821400002E8|
|2010|  2| 2.32810655089999E8|
|2010|  3|1.771853935499992E8|
|2010|  4| 1.92082837389999E8|
|2011|  1| 1.95834838479999E8|
|2011|  2|       1.7647134479E8|
|2011|  3|        2.3487859321E8|
|2011|  4|2.304374437599996E8|
|2012|  1|2.594871966899994E8|
|2012|  2|2.229264593499993E8|
|2012|  3|2.125516714199993E8|
|2012|  4|2.086820953399994E8|
|2013|  1|2.341927744499996E8|
|2013|  2|1.866179134899992E8|
|2013|  3|2.416418492999998E8|
|2013|  4|2.404281667200012E8|
|2014|  1|2.300548111400008E8|
|2014|  2|2.174800714599995E8|
|2014|  3|2.113219669899998E8|
|2014|  4|1.975408959700015E8|
+----+--------+--------------------+
```

User Stories

User Story: Display the number of orders for each item

```
each_item = rdd_df.map(lambda l: (l[2], 1))
item_ocount=each_item.reduceByKey(add).sortBy
(    lambda x: x[1],ascending=False)
```

```
 ## Output:
[('Beverages', 447),
('Fruits', 447),
('Baby Food', 445),
('Cosmetics', 424),
('Household', 424),
('Office Supplies', 420),
('Personal Care', 415),
('Vegetables', 410),
('Meat', 399),
('Snacks', 398),
('Clothes', 386),
('Cereal', 385)]
```

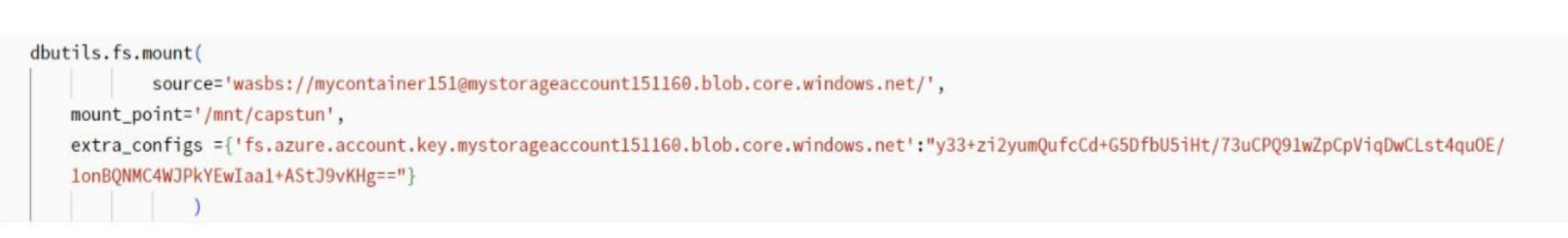
```
item_type_orders_count =
spark.createDataFrame(item_ocount,
['item_type', 'orders_count'])
```

User Story: Display the country with highest sale

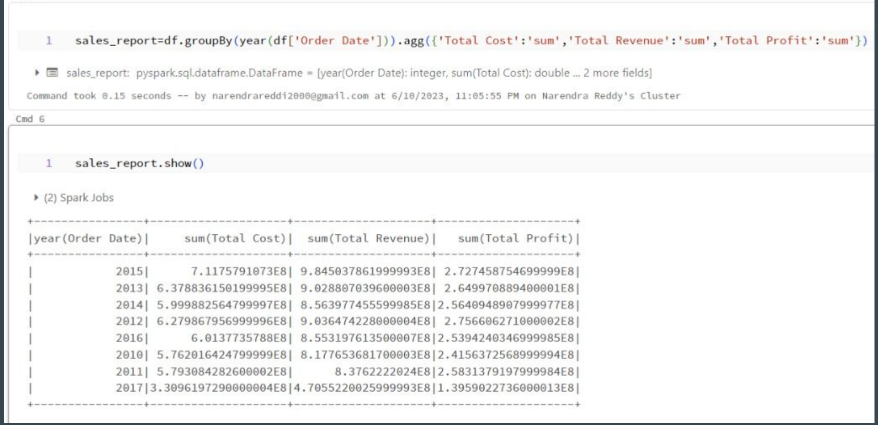
```
country_order = rdd_df.map(lambda l: (l[1], l[11]))
country_revenue=country_order.reduceByKey(add).sortBy(
lambda x: x[1], ascending=False)

countrywise_sales =
spark.createDataFrame(country_revenue, ['country',
'sales'])
```

```
## Output:
+--------------------+--------------------+
| country| sales|
+--------------------+--------------------+
| Rwanda| 6.039873958999999E7|
| Myanmar| 5.883846785E7|
| South Korea| 5.743435542999999E7|
| Ghana| 5.627138265000001E7|
| Niger| 5.529821127999998E7|
| Grenada| 5.498818455000001E7|
|Republic of the C...| 5.437980838999998E7|
| Kosovo| 5.383314279000001E7|
| Czech Republic| 5.354393214E7|
| Ukraine| 5.325231754E7|
| Vanuatu| 5.129172321E7|
```

Databricks



```
dbutils.fs.mount(
            source='wasbs://mycontainer151@mystorageaccount151160.blob.core.windows.net/',
    mount_point='/mnt/capstun',
    extra_configs ={'fs.azure.account.key.mystorageaccount151160.blob.core.windows.net':"y33+zi2yumQufcCd+G5DfbU5iHt/73uCPQ91wZpCpViqDwCLst4quOE/
    1onBQNMC4WJPkYEwIaa1+AStJ9vKHg=="}
                )
```



```
sales_report=df.groupBy(year(df['Order Date'])).agg({'Total Cost':'sum','Total Revenue':'sum','Total Profit':'sum'})
```

sales_report: pyspark.sql.dataframe.DataFrame = [year(Order Date): integer, sum(Total Cost): double ... 2 more fields]

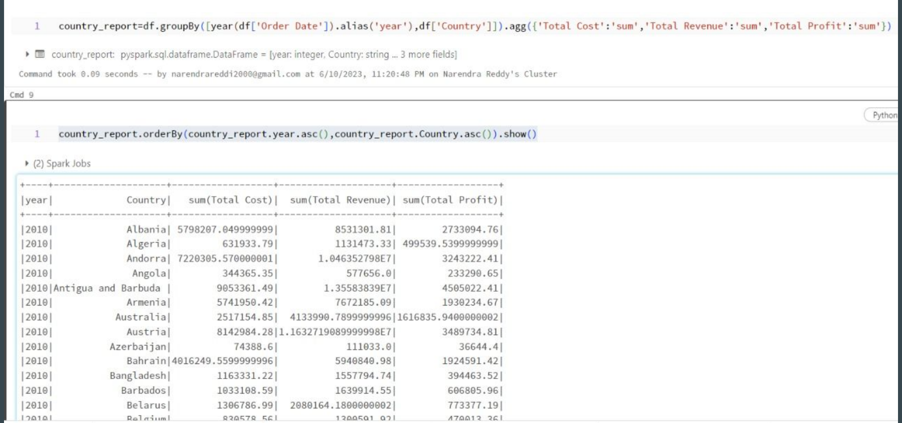
Command took 0.15 seconds -- by narendrareddi2000@gmail.com at 6/10/2023, 11:05:55 PM on Narendra Reddy's Cluster

Cmd 6

```
sales_report.show()
```

(2) Spark Jobs

```
+----------------+-------------------+-------------------+-------------------+
|year(Order Date)|    sum(Total Cost)| sum(Total Revenue)|  sum(Total Profit)|
+----------------+-------------------+-------------------+-------------------+
|            2015|      7.1175791073E8| 9.845037861999993E8| 2.727458754699999E8|
|            2013| 6.378836150199995E8| 9.028807039600003E8| 2.649970889400001E8|
|            2014| 5.999882564799997E8| 8.563977455599985E8|2.5640948907999977E8|
|            2012| 6.279867956999996E8| 9.036474228000004E8| 2.756606271000002E8|
|            2016|      6.0137735788E8| 8.553197613500007E8|2.5394240346999985E8|
|            2010| 5.762016424799999E8| 8.177653681700003E8|2.4156372568999994E8|
|            2011| 5.793084282600002E8|      8.3762222024E8|2.5831379197999984E8|
|            2017|3.3096197290000004E8|4.7055220025999993E8|1.3959022736000013E8|
+----------------+-------------------+-------------------+-------------------+
```



```
country_report=df.groupBy([year(df['Order Date']).alias('year'),df['Country']]).agg({'Total Cost':'sum','Total Revenue':'sum','Total Profit':'sum'})
```

country_report: pyspark.sql.dataframe.DataFrame = [year: integer, Country: string ... 3 more fields]

Command took 0.09 seconds -- by narendrareddi2000@gmail.com at 6/10/2023, 11:20:48 PM on Narendra Reddy's Cluster

Cmd 9

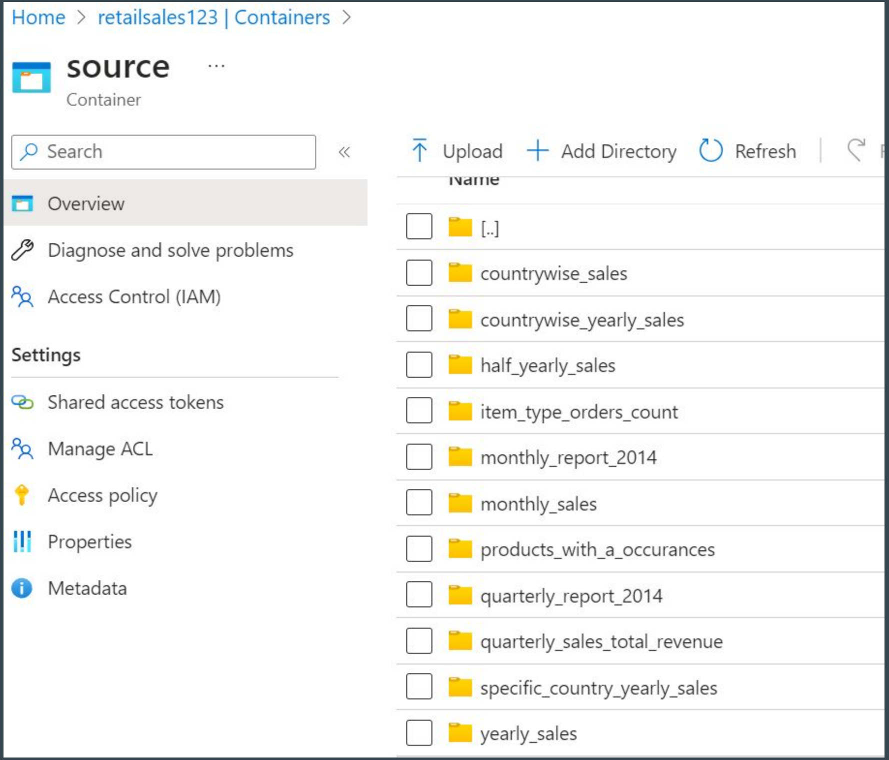
```
country_report.orderBy(country_report.year.asc(),country_report.Country.asc()).show()
```

(2) Spark Jobs

```
+----+--------------------+-----------------+-------------------+------------------+
|year|             Country|  sum(Total Cost)| sum(Total Revenue)| sum(Total Profit)|
+----+--------------------+-----------------+-------------------+------------------+
|2010|             Albania| 5798207.049999999|         8531301.81|        2733094.76|
|2010|             Algeria|        631933.79|         1131473.33| 499539.5399999999|
|2010|             Andorra| 7220305.570000001|      1.046352798E7|        3243222.41|
|2010|              Angola|        344365.35|           577656.0|         233290.65|
|2010|Antigua and Barbuda |       9053361.49|       1.35583839E7|        4505022.41|
|2010|             Armenia|       5741950.42|         7672185.09|        1930234.67|
|2010|           Australia|       2517154.85|  4133990.7899999996|1616835.9400000002|
|2010|             Austria|       8142984.28|1.1632719089999998E7|        3489734.81|
|2010|          Azerbaijan|          74388.6|           111033.0|           36644.4|
|2010|             Bahrain|4016249.5599999996|         5940840.98|        1924591.42|
|2010|          Bangladesh|       1163331.22|         1557794.74|         394463.52|
|2010|            Barbados|       1033108.59|         1639914.55|         606805.96|
|2010|             Belarus|       1306786.99|  2080164.1800000002|         773377.19|
|2010|             Belgium|        830578.56|         1300591.92|         470013.36|
```

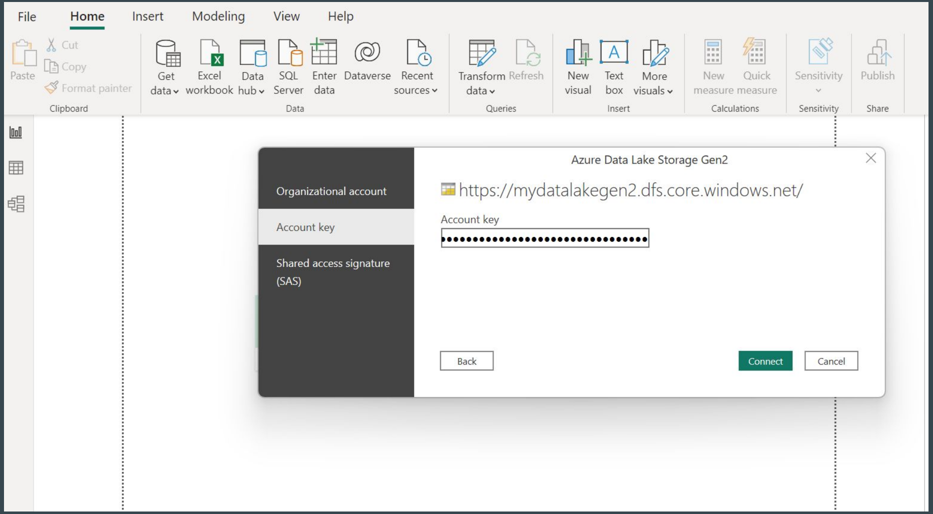
Store data

```
adls_account_name = "retailsales123"
adls_container = "source"
adls_folder = "output"
adls_output_path =
r'abfs://source@retailsales123.dfs.core.w
indows.net/output/countrywise_sales'

# Save the DataFrame to ADLS
countrywise_sales.write \
     .mode("overwrite") \
     .csv(adls_output_path)
```



- □ [..]
- □ countrywise_sales
- □ countrywise_yearly_sales
- □ half_yearly_sales
- □ item_type_orders_count
- □ monthly_report_2014
- □ monthly_sales
- □ products_with_a_occurances
- □ quarterly_report_2014
- □ quarterly_sales_total_revenue
- □ specific_country_yearly_sales
- □ yearly_sales

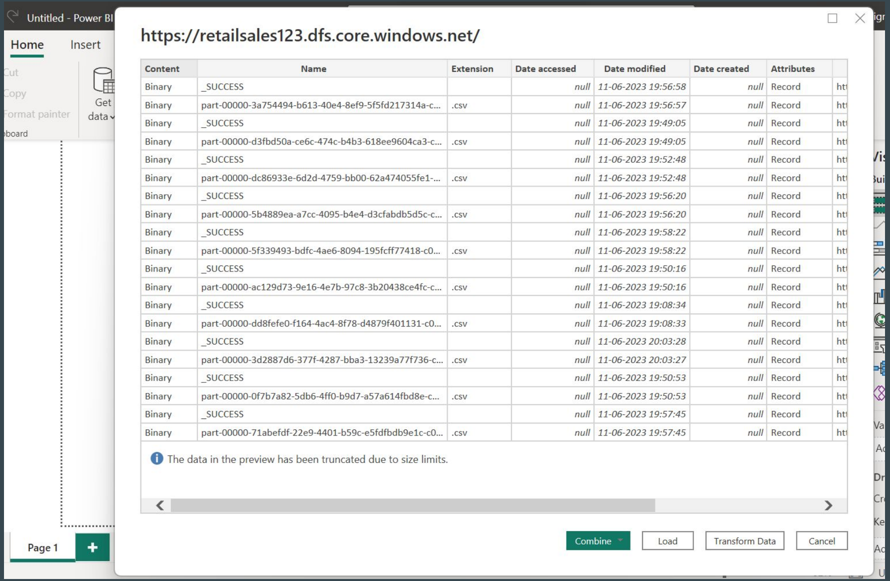
ADLS to PowerBI Desktop Connection



Azure Data Lake Storage Gen2

https://mydatalakegen2.dfs.core.windows.net/

Account key: ••••••••••••••••••••••••••••••••



https://retailsales123.dfs.core.windows.net/

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	
Binary	_SUCCESS		null	11-06-2023 19:56:58	null	Record	ht
Binary	part-00000-3a754494-b613-40e4-8ef9-5f5fd217314a-c...	.csv	null	11-06-2023 19:56:57	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:49:05	null	Record	ht
Binary	part-00000-d3fbd50a-ce6c-474c-b4b3-618ee9604ca3-c...	.csv	null	11-06-2023 19:49:05	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:52:48	null	Record	ht
Binary	part-00000-dc86933e-6d2d-4759-bb00-62a474055fe1-...	.csv	null	11-06-2023 19:52:48	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:56:20	null	Record	ht
Binary	part-00000-5b4889ea-a7cc-4095-b4e4-d3cfabdb5d5c-c...	.csv	null	11-06-2023 19:56:20	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:58:22	null	Record	ht
Binary	part-00000-5f339493-bdfc-4ae6-8094-195fcff77418-c0...	.csv	null	11-06-2023 19:58:22	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:50:16	null	Record	ht
Binary	part-00000-ac129d73-9e16-4e7b-97c8-3b20438ce4fc-c...	.csv	null	11-06-2023 19:50:16	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:08:34	null	Record	ht
Binary	part-00000-dd8fefe0-f164-4ac4-8f78-d4879f401131-c0...	.csv	null	11-06-2023 19:08:33	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 20:03:28	null	Record	ht
Binary	part-00000-3d2887d6-377f-4287-bba3-13239a77f736-c...	.csv	null	11-06-2023 20:03:27	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:50:53	null	Record	ht
Binary	part-00000-0f7b7a82-5db6-4ff0-b9d7-a57a614fbd8e-c...	.csv	null	11-06-2023 19:50:53	null	Record	ht
Binary	_SUCCESS		null	11-06-2023 19:57:45	null	Record	ht
Binary	part-00000-71abefdf-22e9-4401-b59c-e5fdfbdb9e1c-c0...	.csv	null	11-06-2023 19:57:45	null	Record	ht

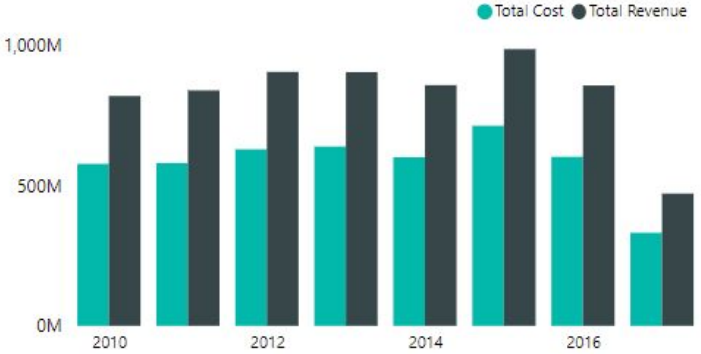
The data in the preview has been truncated due to size limits.

COUNTRY

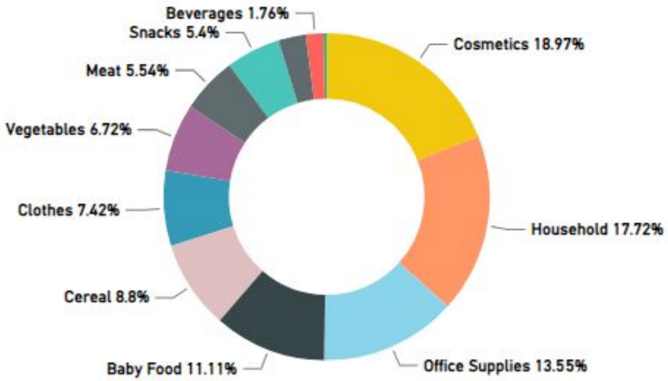
All

RETAIL SALES DASHBOARD

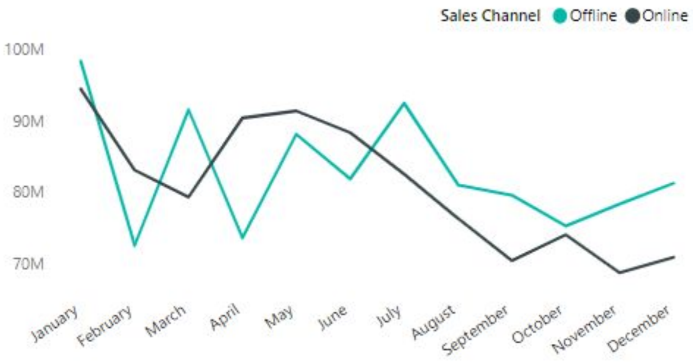
COST VS REVENUE



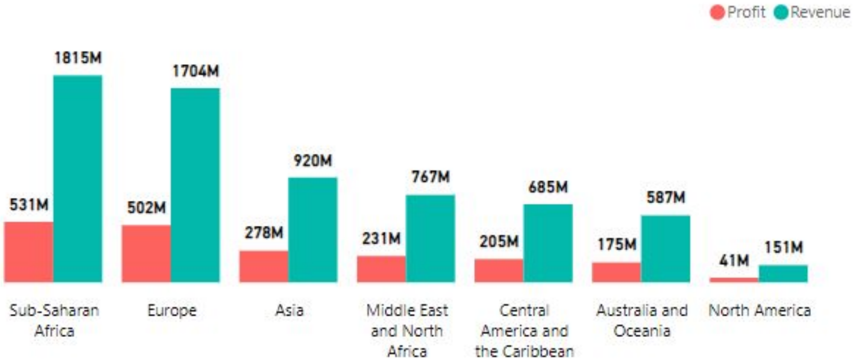
CATEGORYWISE PROFIT



MONTHWISE PROFIT



PROFIT AND REVENUE W.R.T REGION



THANK YOU VENKAT SIR