Mini Project

Big Data Analytics and Visualization Using AWS and PySpark

Submitted by: Varunkumar Sonawane (vsonawa@iu.edu)

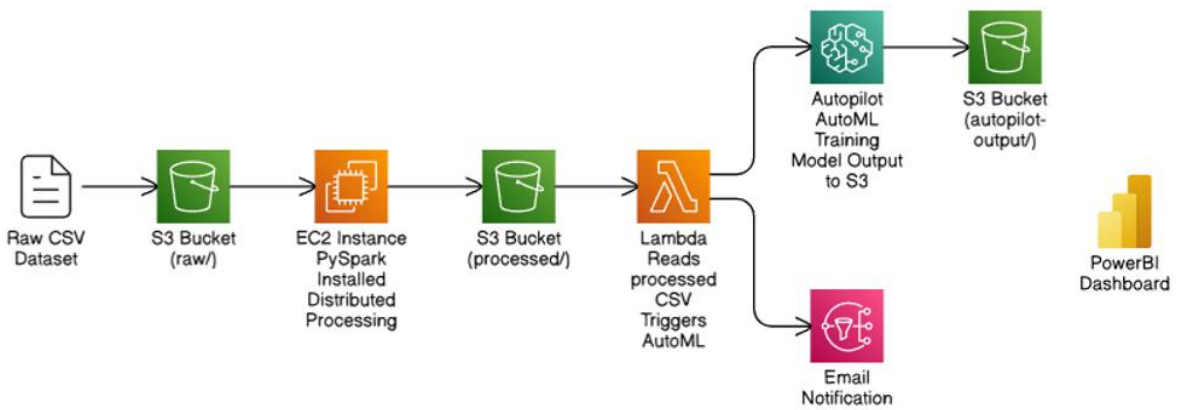
Introduction

This project focuses on building a complete big data analytics pipeline using AWS services and a Linux based PySpark environment. The **goal** of the project is to efficiently process large scale health data, engineer meaningful features, perform distributed analysis, and automate machine learning model training using cloud-based tools. The objective is to demonstrate practical data engineering skills, scalable processing, and automated predictive analytics in a real-world setting.

The **dataset** used in this project contains one hundred thousand records with twenty-one health and lifestyle attributes, including BMI, cholesterol levels, physical activity, sleep duration, blood pressure, calorie intake, and various risk related behaviors. The diversity of the data makes it suitable for cleaning, transformation, aggregation, SQL based exploration, and disease risk prediction.

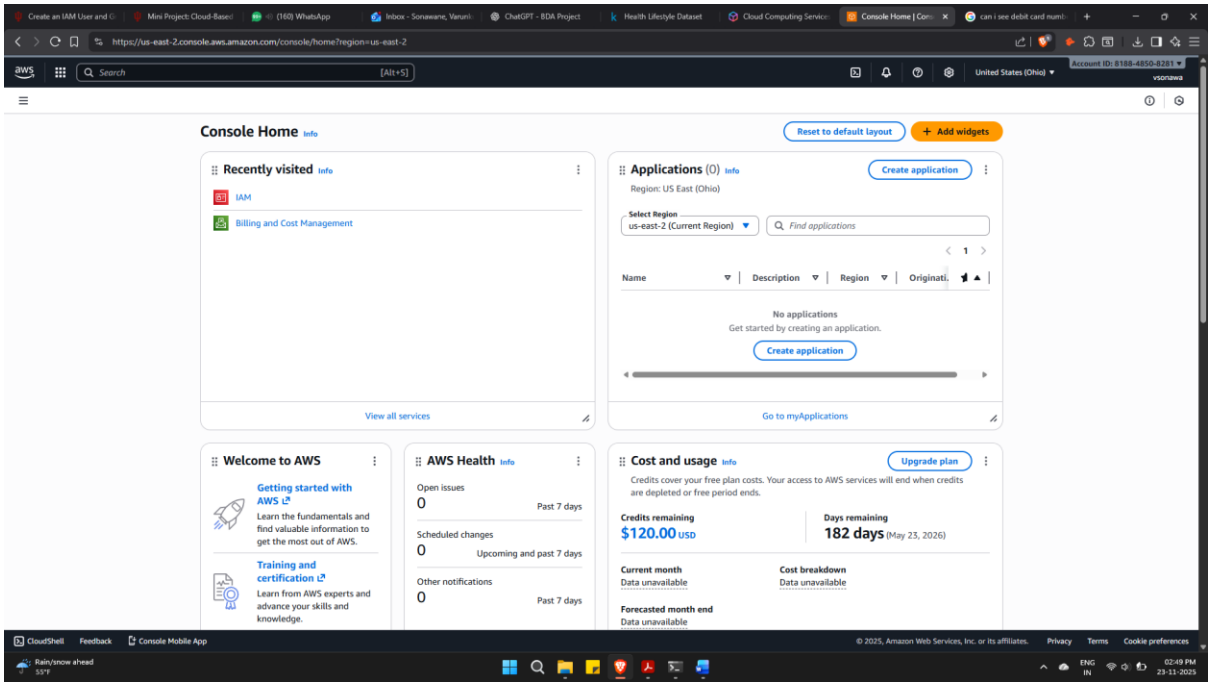
Amazon S3 is used for storing raw and processed data, while an EC2 instance running PySpark handles ingestion and distributed processing. Feature engineering and health metric calculations are performed using PySpark, followed by machine learning with AWS SageMaker Autopilot to automatically build and evaluate models. Power BI Desktop is used to create an interactive dashboard for visualization. The project also includes a fully automated serverless pipeline using S3 triggers, AWS Lambda, SageMaker, and SNS for continuous data processing and AutoML.

The **complete architecture diagram**, showing both the PySpark workflow and the automated pipeline, should be placed **immediately after this introduction** to provide a clear visual overview of the system before moving into the methodology section.

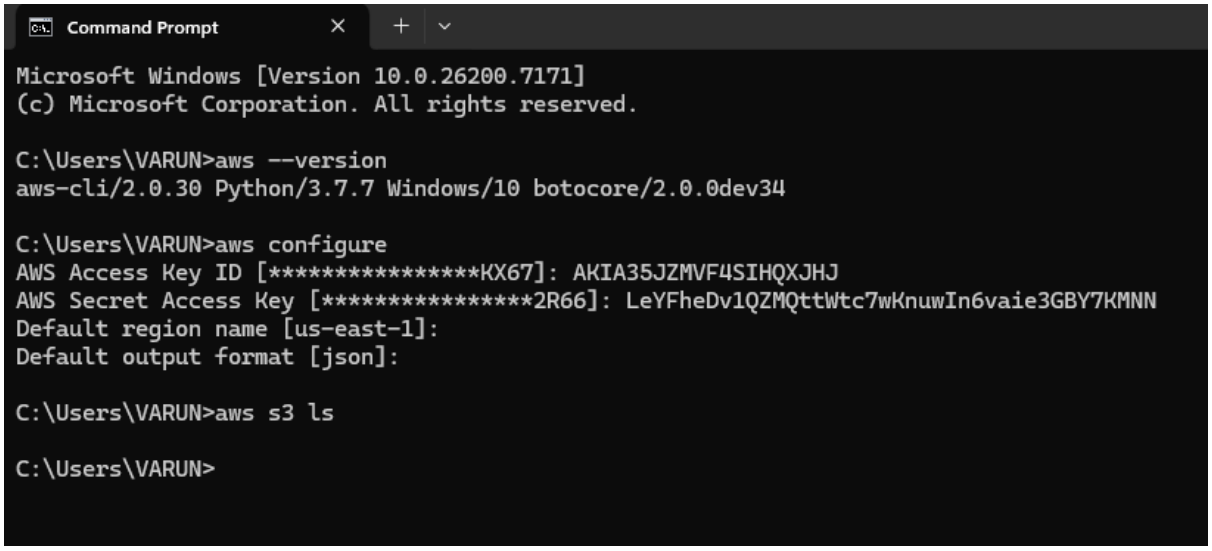


Methodology

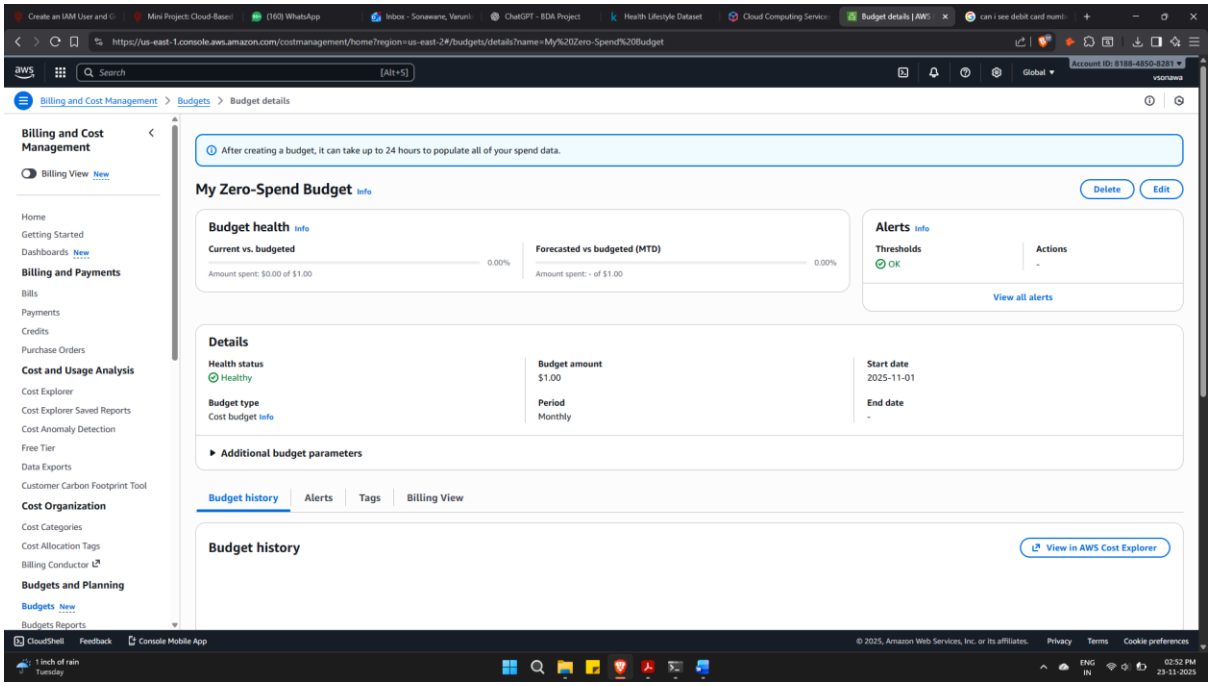
1.1 Creating AWS account



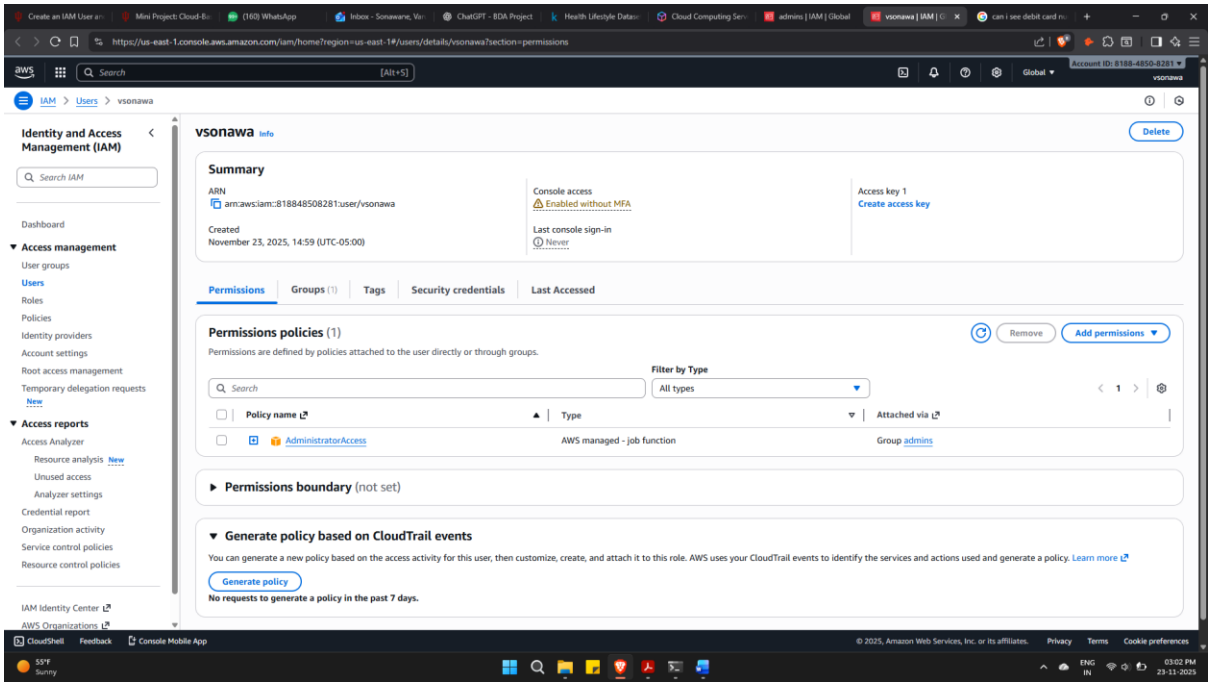
1.2 Installing and Configuring AWS CLI



1.3 Creating Billing alert (Billing and Cost Management)



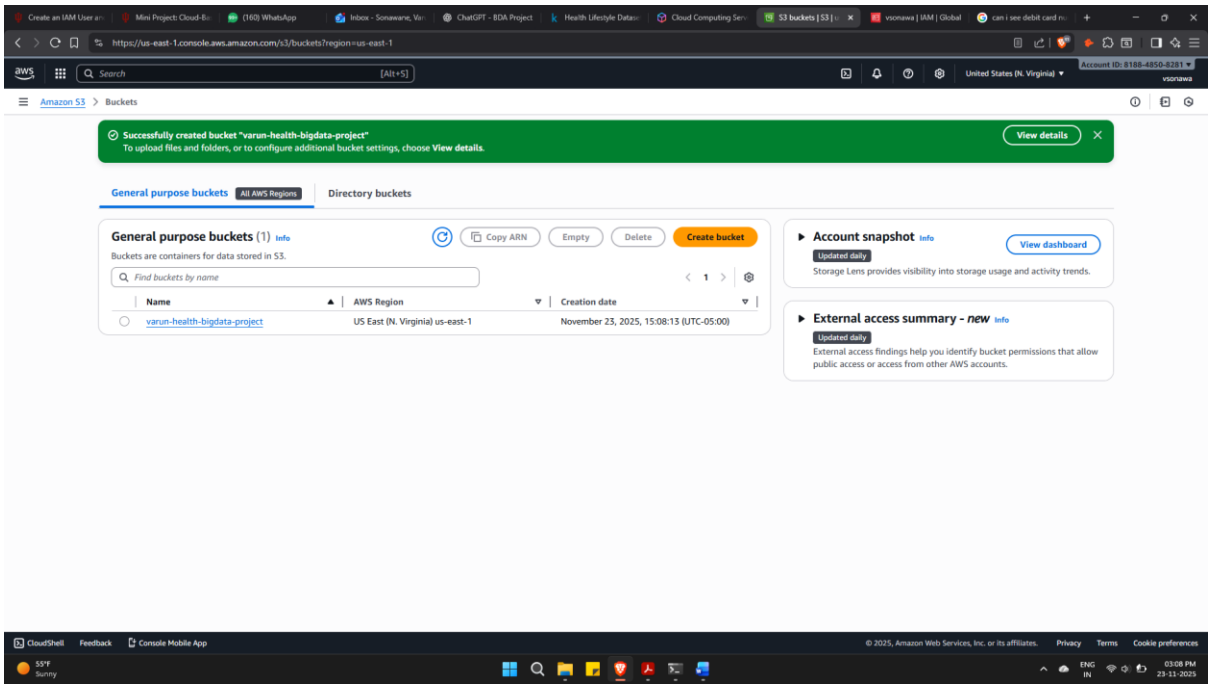
1.4 Creating IAM user (vsonawa) and User group(admins)



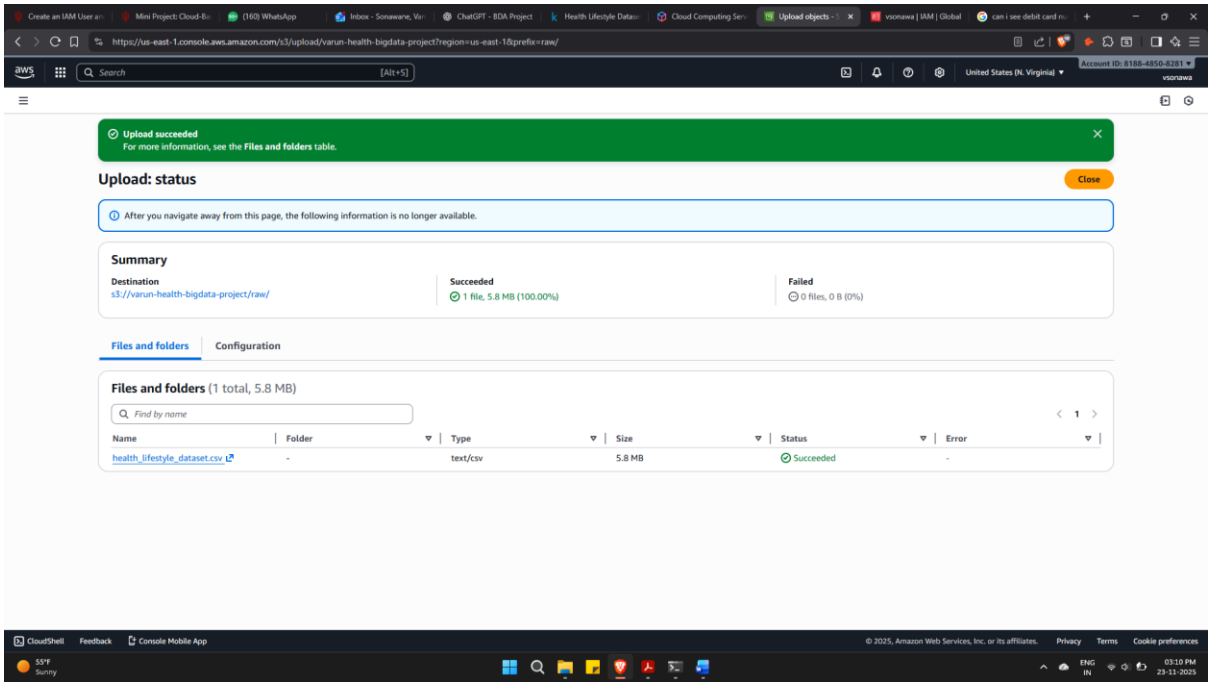
Environment Setup

1. **AWS S3 for Data Storage**

a. **Step 1: Create an S3 bucket to store both raw and processed data.**



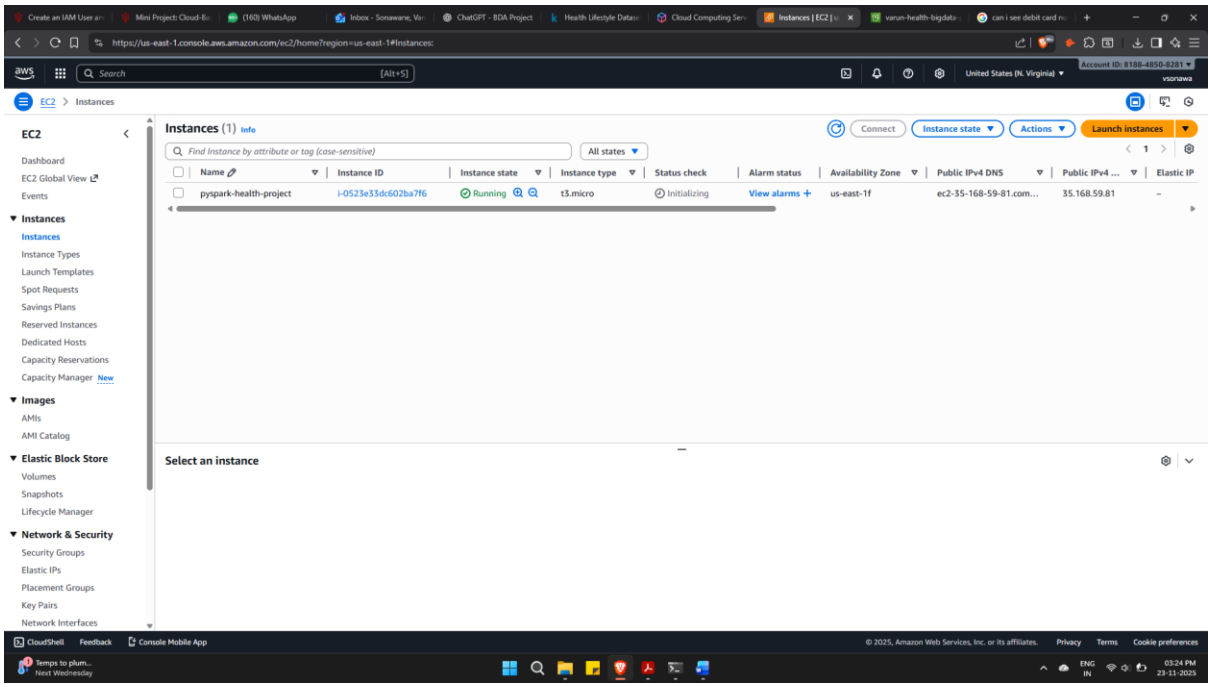
b. Step 2: Upload the raw dataset to the S3 bucket.



2. Linux Environment with PySpark

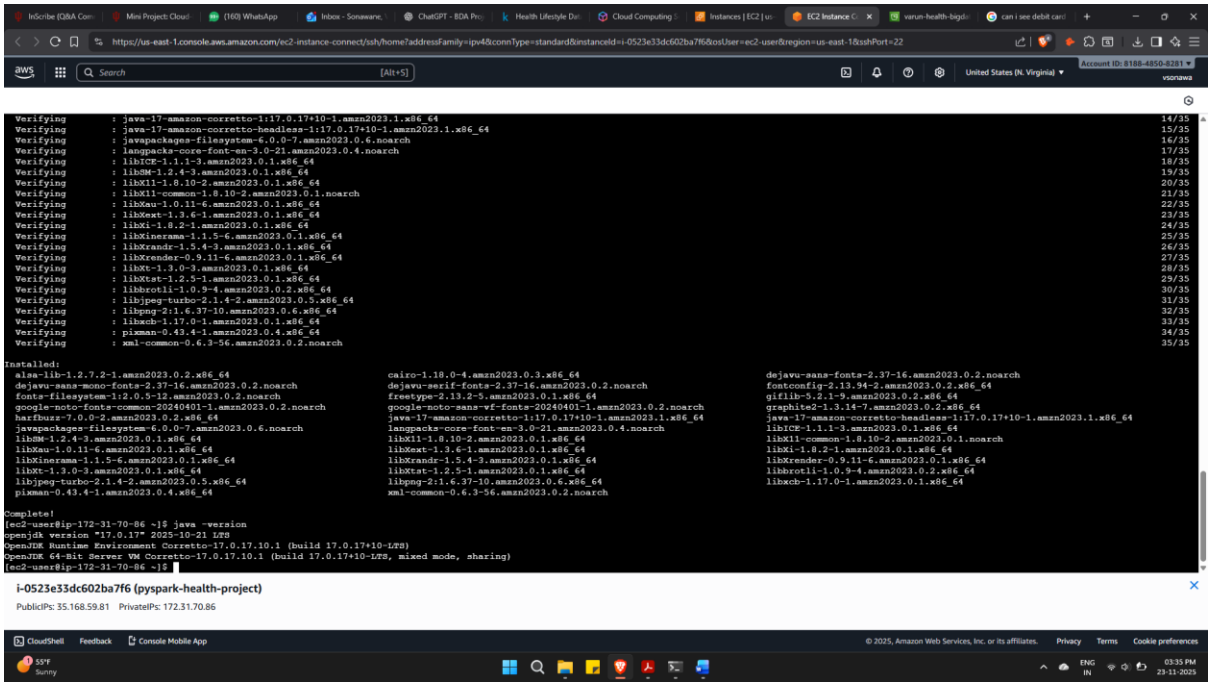
a. Step 1: Set up a Linux-based environment, either locally or using an AWS EC2 instance.

(instance name : pyspark-health-project)

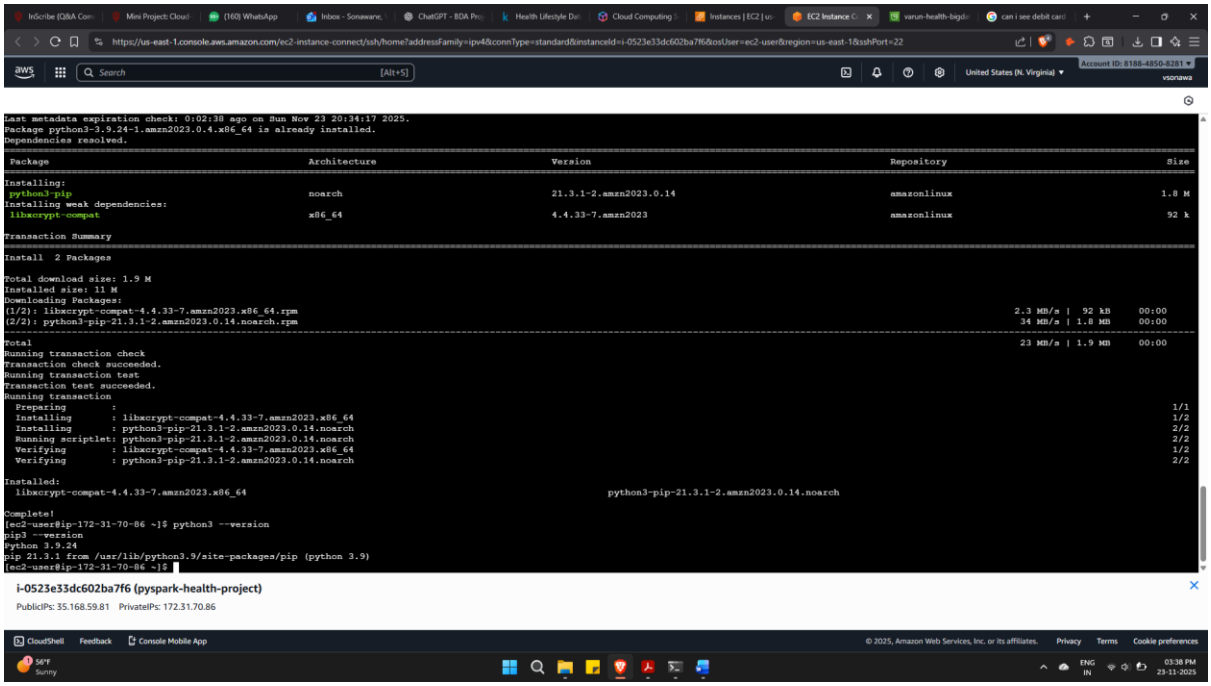


b. Step 2: Install PySpark for distributed data processing.

1. Installing Java (required for spark)



2. Installing Python, pip



3. Installing Pyspark

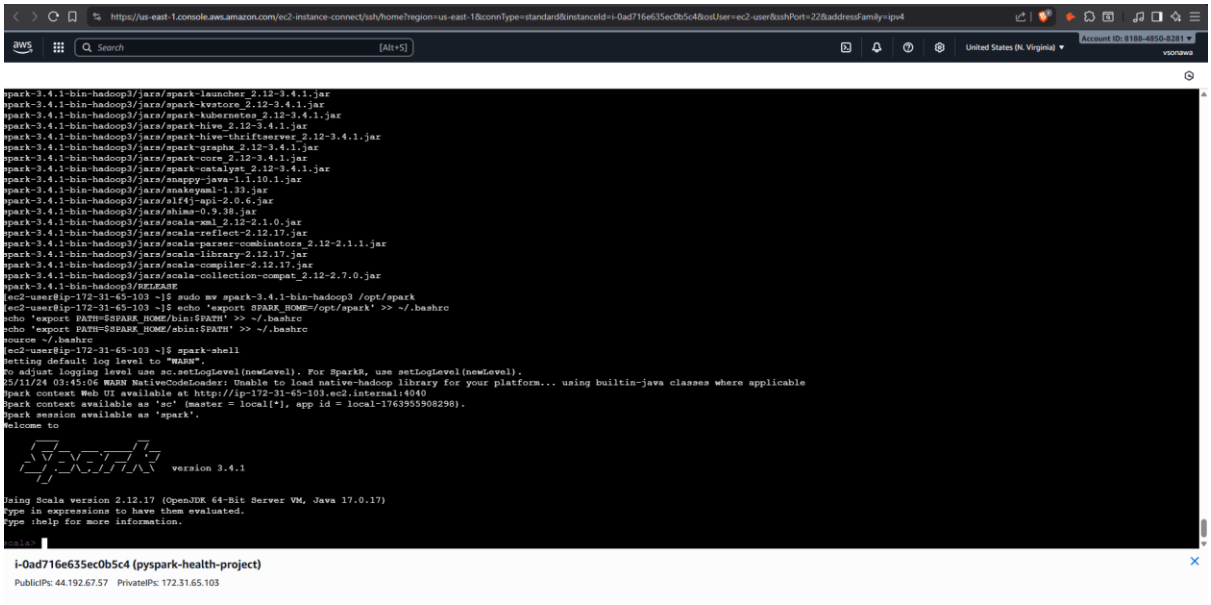
Step 1: Install wget and tar if missing
sudo yum install wget -y

Step 2: Download Apache Spark
wget https://archive.apache.org/dist/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3.tgz

Step 3: Extract the file
tar xvf spark-3.4.1-bin-hadoop3.tgz

Step 4: Move Spark to a permanent location
sudo mv spark-3.4.1-bin-hadoop3 /opt/spark

Step 5: Add Spark to PATH
echo 'export SPARK_HOME=/opt/spark' >> ~/.bashrc
echo 'export PATH=$SPARK_HOME/bin:$PATH' >> ~/.bashrc
echo 'export PATH=$SPARK_HOME/sbin:$PATH' >> ~/.bashrc
source ~/.bashrc



c. Step 3: Configure AWS CLI to interact with S3 buckets

```
[ec2-user@ip-172-31-73-97 ~]$ sudo yum install awscli -y
Last metadata expiration check: 0:03:03 ago on Mon Nov 24 04:10:44 2025.
Package awscli-2-2.30.4-1.amzn2023.0.1.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-73-97 ~]$ aws configure
AWS Access Key ID [None]: AKIA35JZMVF4SIHQXJHJ
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
[ec2-user@ip-172-31-73-97 ~]$
```

Data Pipeline Tasks

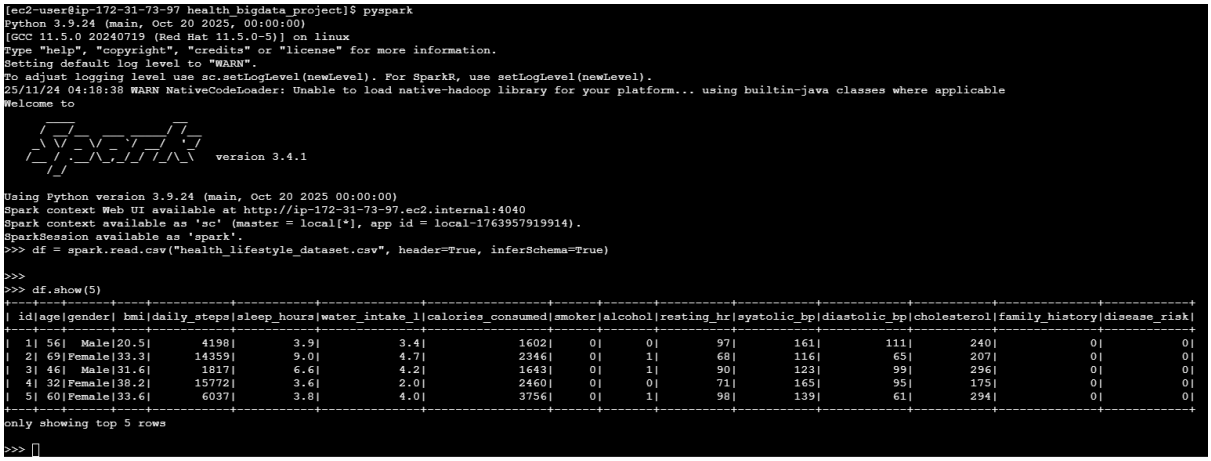
Task 1: Data Ingestion from S3

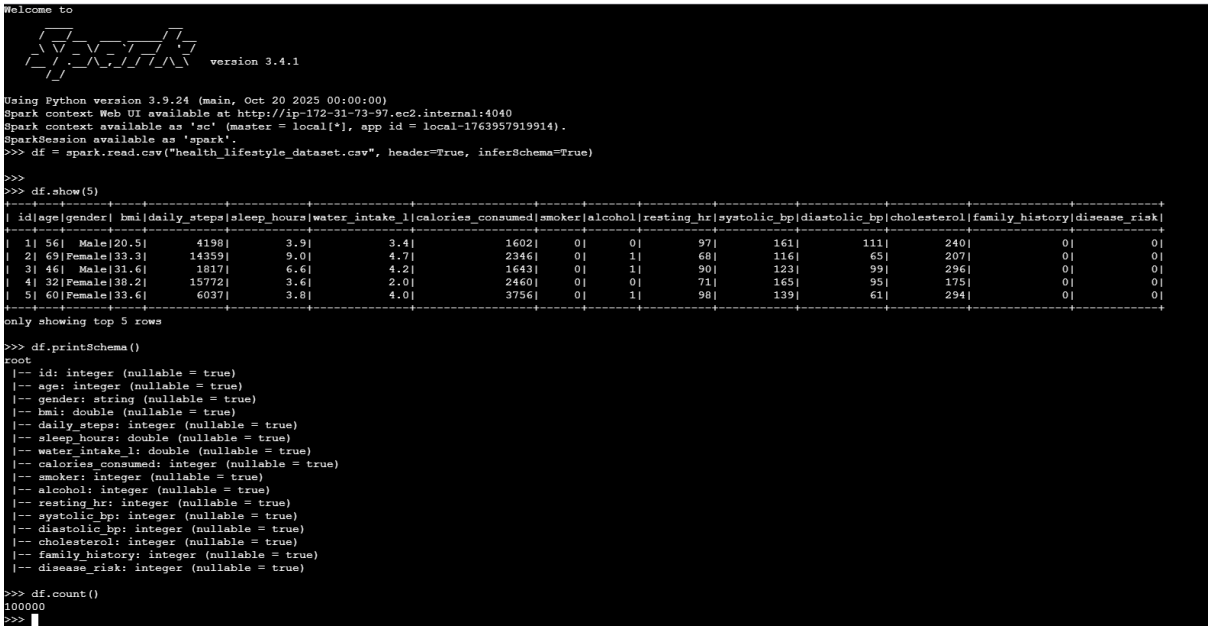
Steps:

1. Use AWS CLI or PySpark's built-in S3 support to load the dataset directly.

```
[ec2-user@ip-172-31-73-97 ~]$ mkdir ~/health_bigdata_project
cd ~/health_bigdata_project
[ec2-user@ip-172-31-73-97 health_bigdata_project]$ aws s3 cp s3://varun-health-bigdata-project/raw/health_lifestyle_dataset.csv .
download: s3://varun-health-bigdata-project/raw/health_lifestyle_dataset.csv to ./health_lifestyle_dataset.csv
[ec2-user@ip-172-31-73-97 health_bigdata_project]$ ls
health_lifestyle_dataset.csv
```

2. Confirm successful ingestion by inspecting the dataset.





Task 2: Data Processing with PySpark

1. **Data Transformation:** Create at least 2 new columns (e.g., Year , Month) to aid in analysis.

PySpark preprocessing code

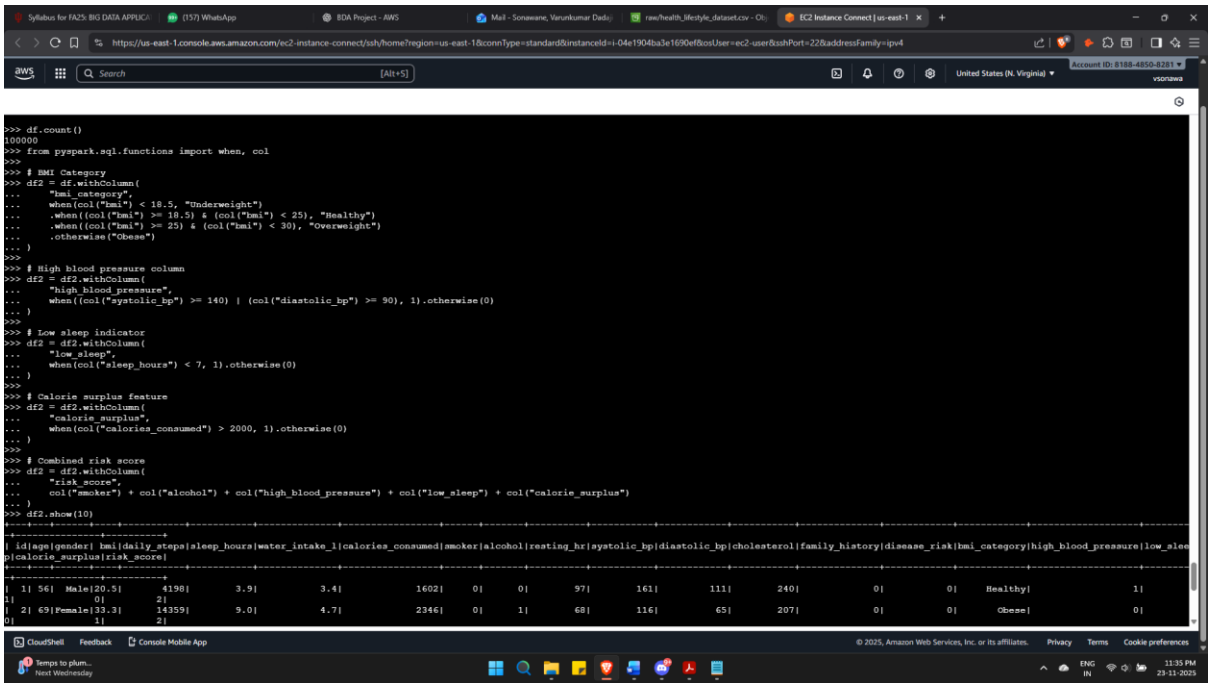
```
from pyspark.sql.functions import when, col
# BMI Category
df2 = df.withColumn(
   "bmi_category",
   when(col("bmi") < 18.5, "Underweight")
   .when((col("bmi") >= 18.5) & (col("bmi") < 25), "Healthy")
   .when((col("bmi") >= 25) & (col("bmi") < 30), "Overweight")
   .otherwise("Obese")
)
```

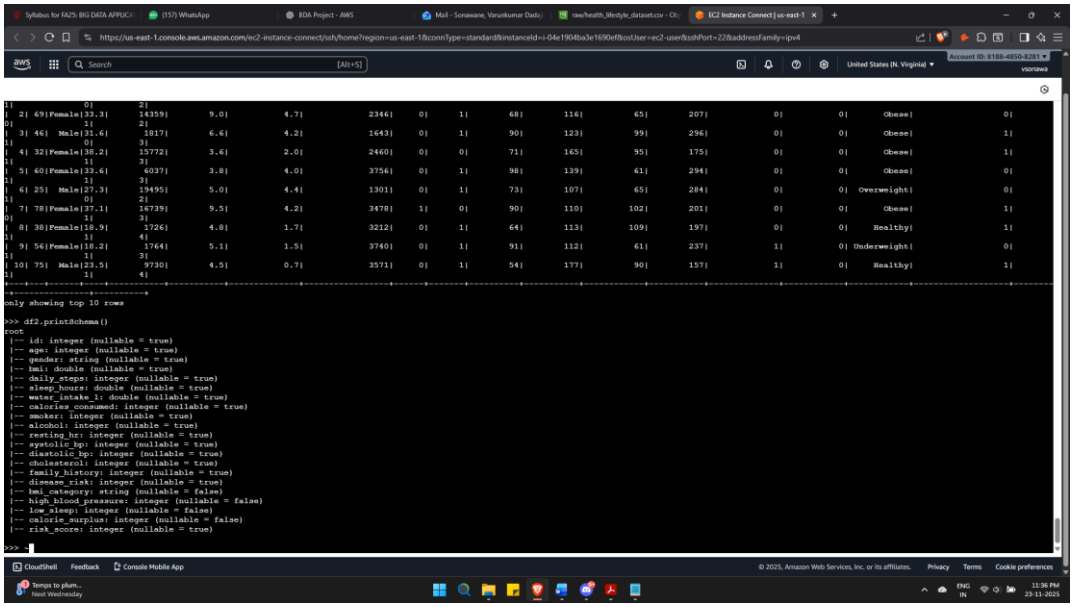
```
# High blood pressure column
df2 = df2.withColumn(
   "high_blood_pressure",
   when((col("systolic_bp") >= 140) | (col("diastolic_bp") >= 90), 1).otherwise(0)
)

# Low sleep indicator
df2 = df2.withColumn(
   "low_sleep",
   when(col("sleep_hours") < 7, 1).otherwise(0)
)

# Calorie surplus feature
df2 = df2.withColumn(
   "calorie_surplus",
   when(col("calories_consumed") > 2000, 1).otherwise(0)
)

# Combined risk score
df2 = df2.withColumn(
   "risk_score",
   col("smoker") + col("alcohol") + col("high_blood_pressure") + col("low_sleep") +
col("calorie_surplus")
)
```

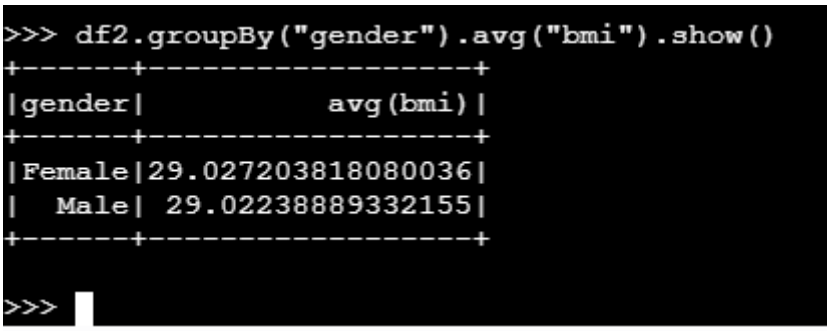




2. **Data Aggregation:** Compute at least 5 key metrics, such as Total revenue by region, Monthly spending trends, Top 10 customers by transaction value (These metrics may vary depending on the specifics of your dataset, but the goal is to aggregate the data in ways that enable meaningful analysis and decision-making.)

Aggregation 1: Average BMI by gender
This tells us which gender has higher BMI on average.

df2.groupBy("gender").avg("bmi").show()



Aggregation 2: Average daily steps by age group
Create age bands and find physical activity patterns.
from pyspark.sql.functions import when

```
df3 = df2.withColumn(
   "age_group",
   when(col("age") < 30, "Under 30")
   .when(col("age") < 50, "30 to 49")
   .when(col("age") < 65, "50 to 64")
   .otherwise("65 plus")
)
```

df3.groupBy("age_group").avg("daily_steps").orderBy("age_group").show()

```
>>> from pyspark.sql.functions import when
>>>
>>> df3 = df2.withColumn(
...     "age_group",
...     when(col("age") < 30, "Under 30")
...     .when(col("age") < 50, "30 to 49")
...     .when(col("age") < 65, "50 to 64")
...     .otherwise("65 plus")
... )
>>>
>>> df3.groupBy("age_group").avg("daily_steps").orderBy("age_group").show()
+---------+------------------+
|age_group|  avg(daily_steps)|
+---------+------------------+
| 30 to 49|10474.393635972725|
| 50 to 64|10508.087927657623|
|  65 plus|10509.322770484381|
| Under 30|10416.436905881124|
+---------+------------------+
```

Aggregation 3: Percentage of high blood pressure cases
Useful health metric.
df2.groupBy("high_blood_pressure").count().show()

```
>>> df2.groupBy("high_blood_pressure").count().show()
+-------------------+-----+
|high_blood_pressure|count|
+-------------------+-----+
|                  1|72264|
|                  0|27736|
+-------------------+-----+
```

Aggregation 4: Average sleep hours by disease risk
Shows correlation between sleep and disease.
df2.groupBy("disease_risk").avg("sleep_hours").show()

```
>>> df2.groupBy("disease_risk").avg("sleep_hours").show()
+------------+------------------+
|disease_risk|  avg(sleep_hours)|
+------------+------------------+
|           1| 6.500870230852913|
|           0|6.4887841019432635|
+------------+------------------+
```

Aggregation 5: Average cholesterol by BMI category
This is meaningful for medical analysis.
df2.groupBy("bmi_category").avg("cholesterol").show()

```
>>> df2.groupBy("bmi_category").avg("cholesterol").show()
+------------+------------------+
|bmi_category|  avg(cholesterol)|
+------------+------------------+
|  Overweight| 224.0055787392928|
| Underweight|223.36733668341708|
|       Obese|224.44890304008374|
|     Healthy| 224.3610288863938|
+------------+------------------+
```

Aggregation 6: Top 10 highest risk individuals

df2.orderBy(col("risk_score").desc()).show(10)

```
>>> df2.orderBy(col("risk_score").desc()).show(10)
```

id	age	gender	bmi	daily_steps	sleep_hours	water_intake_l	calories_consumed	smoker	alcohol	resting_hr	systolic_bp	diastolic_bp	cholesterol	family_history	disease_risk	bmi_category	high_blood_pressure	low_sleep	calorie_surplus	risk_score
84831	39	Male	23.9	19261	5.5	4.4	3216	1	1	84	124	110	237	0	0	Healthy	1	1	1	5
161	65	Male	19.6	12853	3.8	2.6	3111	1	1	91	170	102	299	0	1	Healthy	1	1	1	5
85072	34	Female	25.6	15909	3.8	4.0	2374	1	1	81	101	116	252	0	0	Overweight	1	1	1	5
789	24	Male	21.6	1904	4.6	3.8	2365	1	1	90	95	102	199	0	0	Healthy	1	1	1	5
84837	42	Female	23.6	10952	5.2	2.6	3983	1	1	72	132	119	215	0	0	Healthy	1	1	1	5
229	47	Female	35.7	4274	5.8	2.3	2648	1	1	82	146	96	240	0	0	Obese	1	1	1	5
84633	43	Female	29.9	6131	6.3	4.8	3819	1	1	87	117	94	199	0	0	Overweight	1	1	1	5
16	70	Male	25.8	18483	5.6	1.4	2759	1	1	86	153	105	167	0	1	Overweight	1	1	1	5
84839	63	Male	25.4	19844	6.0	3.0	3250	1	1	83	143	112	256	1	1	Overweight	1	1	1	5
327	20	Male	21.5	9413	3.9	3.5	3245	1	1	84	158	68	255	0	0	Healthy	1	1	1	5

only showing top 10 rows

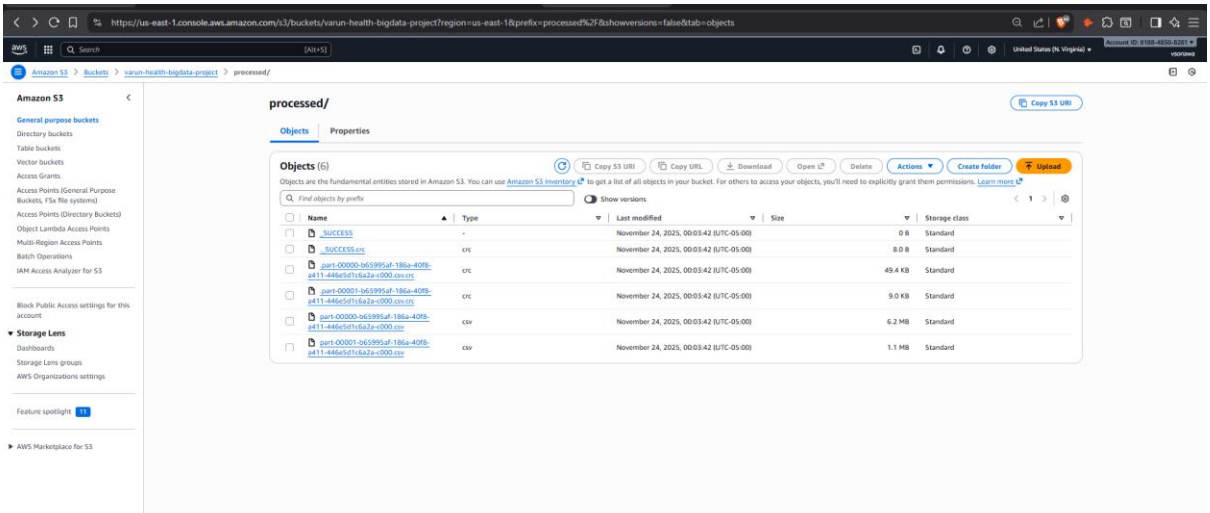
Task 3: Store Processed Data Back to S3

Steps: 1. Export data in CSV or Parquet format.

2. Upload the processed data to a designated S3 location for easy access.

```
>>> df2.write.mode("overwrite").option("header", "true").csv("processed_output")
>>> exit()
[ec2-user@ip-172-31-73-97 health_bigdata_project]$ aws s3 cp processed_output s3://varun-health-bigdata-project/processed/ --recursive
upload: processed_output/._SUCCESS.crc to s3://varun-health-bigdata-project/processed/._SUCCESS.crc
upload: processed_output/_SUCCESS to s3://varun-health-bigdata-project/processed/_SUCCESS
upload: processed_output/.part-00001-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv.crc to s3://varun-health-bigdata-project/processed/.part-00001-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv.crc
upload: processed_output/.part-00000-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv.crc to s3://varun-health-bigdata-project/processed/.part-00000-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv.crc
upload: processed_output/part-00000-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv to s3://varun-health-bigdata-project/processed/part-00000-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv
upload: processed_output/part-00001-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv to s3://varun-health-bigdata-project/processed/part-00001-b65995af-186a-40f8-a411-446e5d1c6a2a-c000.csv
[ec2-user@ip-172-31-73-97 health_bigdata_project]$
```

```
>>> df2.coalesce(1).write.mode("overwrite").option("header", "true").csv("permanent_processed")
>>> exit()
[ec2-user@ip-172-31-73-97 health_bigdata_project]$ cd ~/health_bigdata_project/permanent_processed
[ec2-user@ip-172-31-73-97 permanent_processed]$ ls
_SUCCESS  part-00000-103fc401-4100-4ee9-b43b-aa75bb29aedf-c000.csv
[ec2-user@ip-172-31-73-97 permanent_processed]$
```



Task 4: Data Analysis Using Spark SQL
Objective: Use SQL to derive insights (Atleast 5 Queries).
Query 1: Average BMI by gender

```
spark.sql("""
SELECT gender, AVG(bmi) AS avg_bmi
FROM health
GROUP BY gender
""").show()
```

Insight: Helps understand BMI difference across genders.

```
>>> spark.sql("""
... SELECT gender, AVG(bmi) AS avg_bmi
... FROM health
... GROUP BY gender
... """).show()
+------+------------------+
|gender|           avg_bmi|
+------+------------------+
|Female|29.027203818080036|
|  Male| 29.02238889332155|
+------+------------------+
```

Query 2: Average daily steps by age group

First create an age group:

```
spark.sql("""
SELECT
  CASE
    WHEN age < 30 THEN 'Under 30'
    WHEN age BETWEEN 30 AND 49 THEN '30 to 49'
    WHEN age BETWEEN 50 AND 64 THEN '50 to 64'
    ELSE '65 plus'
  END AS age_group,
  AVG(daily_steps) AS avg_steps
FROM health
GROUP BY age_group
ORDER BY age_group
""").show()
```

Insight: Shows physical activity trends.

```
>>> spark.sql("""
... SELECT
...     CASE
...         WHEN age < 30 THEN 'Under 30'
...         WHEN age BETWEEN 30 AND 49 THEN '30 to 49'
...         WHEN age BETWEEN 50 AND 64 THEN '50 to 64'
...         ELSE '65 plus'
...     END AS age_group,
...     AVG(daily_steps) AS avg_steps
... FROM health
... GROUP BY age_group
... ORDER BY age_group
... """).show()
+---------+------------------+
|age_group|         avg_steps|
+---------+------------------+
| 30 to 49|10474.393635972725|
| 50 to 64|10508.087927657623|
|  65 plus|10509.322770484381|
| Under 30|10416.436905881124|
+---------+------------------+
```

Query 3: Count of high blood pressure cases

```
spark.sql("""
SELECT high_blood_pressure, COUNT(*) AS total_people
FROM health
GROUP BY high_blood_pressure
""").show()
```

Insight: How common high BP is in the dataset.

```
>>> spark.sql("""
... SELECT high_blood_pressure, COUNT(*) AS total_people
... FROM health
... GROUP BY high_blood_pressure
... """).show()
+-------------------+------------+
|high_blood_pressure|total_people|
+-------------------+------------+
|                  1|       72264|
|                  0|       27736|
+-------------------+------------+
```

Query 4: Average sleep hours by disease risk

```
spark.sql("""
SELECT disease_risk, AVG(sleep_hours) AS avg_sleep
FROM health
GROUP BY disease_risk
""").show()
```

Insight: Poor sleep often correlates with higher disease risk.

```
>>> spark.sql("""
... SELECT disease_risk, AVG(sleep_hours) AS avg_sleep
... FROM health
... GROUP BY disease_risk
... """).show()
+------------+------------------+
|disease_risk|         avg_sleep|
+------------+------------------+
|           1| 6.500870230852913|
|           0|6.4887841019432635|
+------------+------------------+
```

Query 5: Average cholesterol by BMI category

```
spark.sql("""
SELECT bmi_category, AVG(cholesterol) AS avg_cholesterol
FROM health
GROUP BY bmi_category
ORDER BY avg_cholesterol DESC
""").show()
```

Insight: High BMI categories often have higher cholesterol.

```
>>> spark.sql("""
... SELECT bmi_category, AVG(cholesterol) AS avg_cholesterol
... FROM health
... GROUP BY bmi_category
... ORDER BY avg_cholesterol DESC
... """).show()
+------------+------------------+
|bmi_category|   avg_cholesterol|
+------------+------------------+
|       Obese|224.44890304008374|
|     Healthy| 224.3610288863938|
|  Overweight| 224.0055787392928|
| Underweight|223.36733668341708|
+------------+------------------+
```

Query 6: Top 10 highest lifestyle risk individuals

```
spark.sql("""
SELECT id, age, gender, risk_score
FROM health
ORDER BY risk_score DESC
LIMIT 10
""").show()
```

```
>>> spark.sql("""
... SELECT id, age, gender, risk_score
... FROM health
... ORDER BY risk_score DESC
... LIMIT 10
... """).show()
+-----+---+------+----------+
|   id|age|gender|risk_score|
+-----+---+------+----------+
|84831| 39|  Male|         5|
|  161| 65|  Male|         5|
|85011| 20|  Male|         5|
|  779| 76|Female|         5|
|84837| 42|Female|         5|
|  229| 47|Female|         5|
|84619| 41|  Male|         5|
|   48| 64|Female|         5|
|84839| 63|  Male|         5|
|  327| 20|  Male|         5|
+-----+---+------+----------+
```

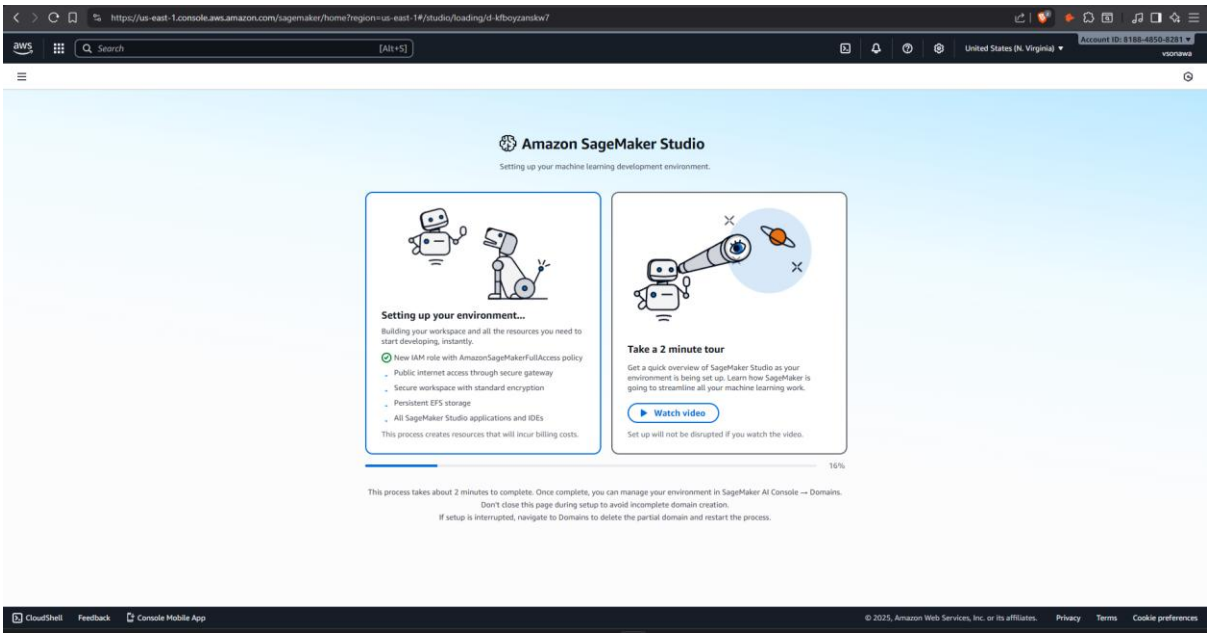
Query 7: Correlation check between activity and BMI

spark.sql("""
SELECT CORR(daily_steps, bmi) AS steps_bmi_correlation
FROM health
""").show()

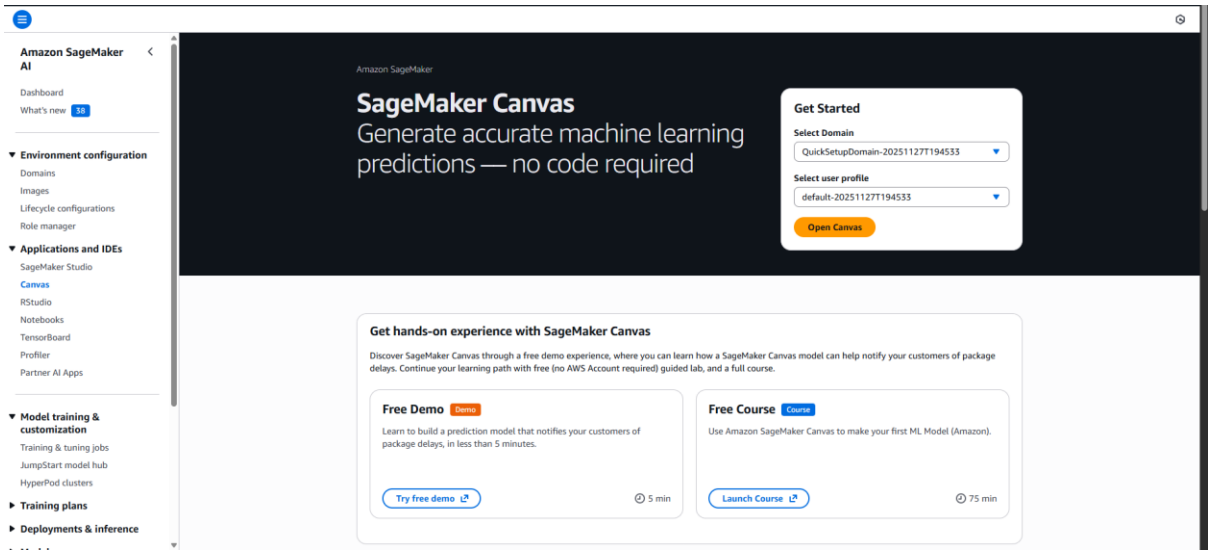
```
>>> spark.sql("""
... SELECT CORR(daily_steps, bmi) AS steps_bmi_correlation
... FROM health
... """).show()
+--------------------+
|steps_bmi_correlation|
+--------------------+
| 0.0018140091715289 51|
+--------------------+
```

Task 5: Machine Learning with AWS SageMaker Autopilot

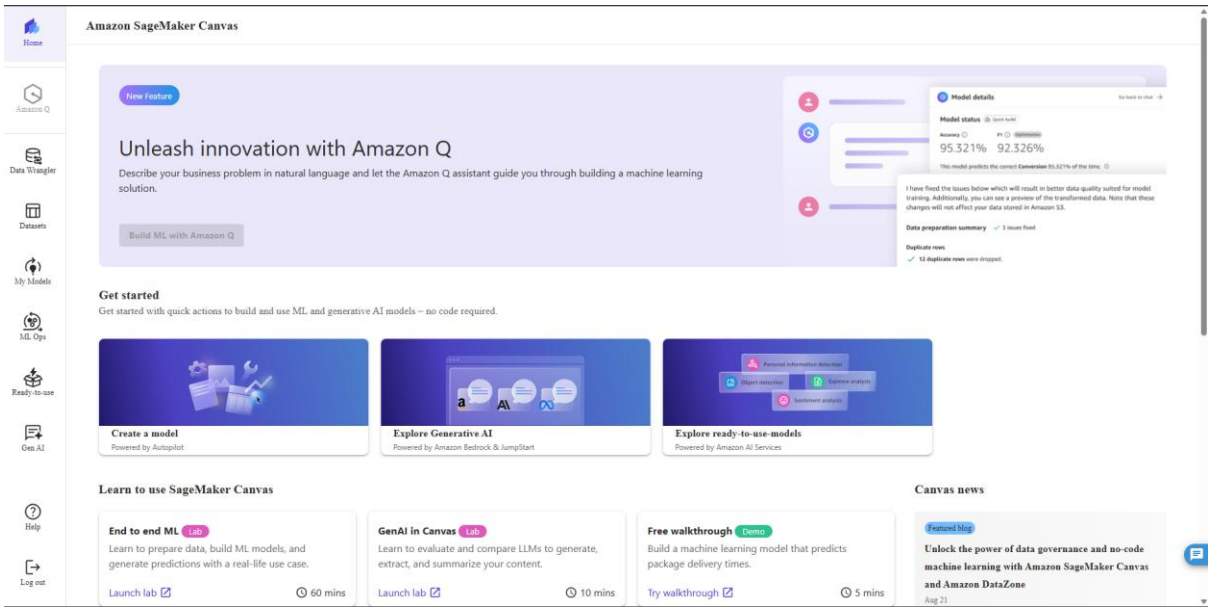
1. Creating a Domain in Sagemaker



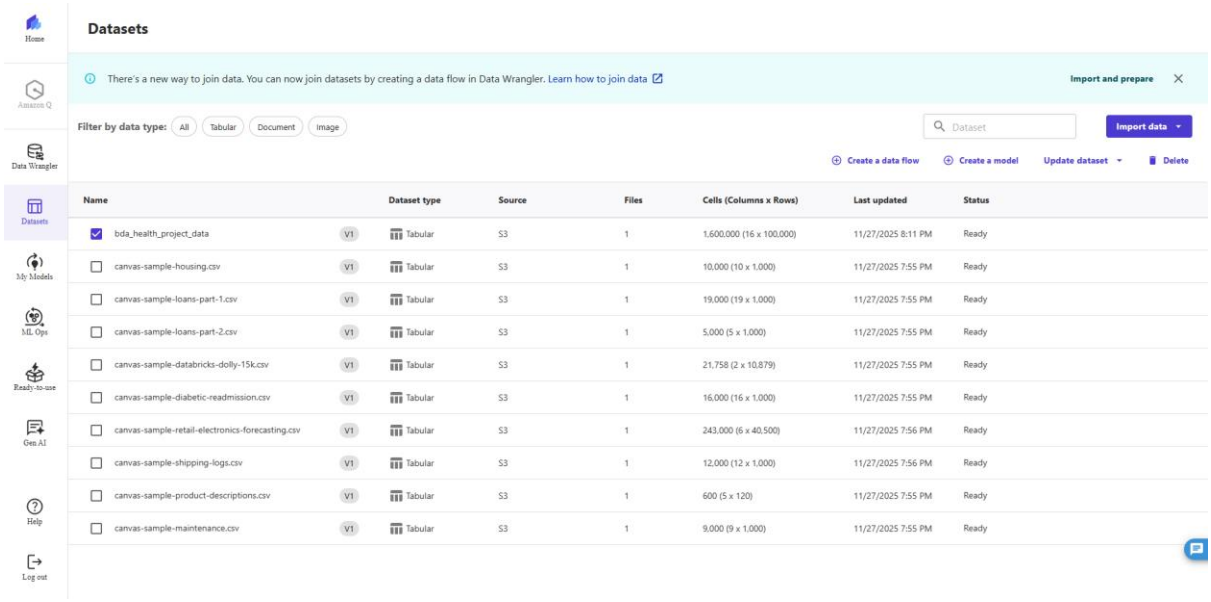
2. Opening Canvas



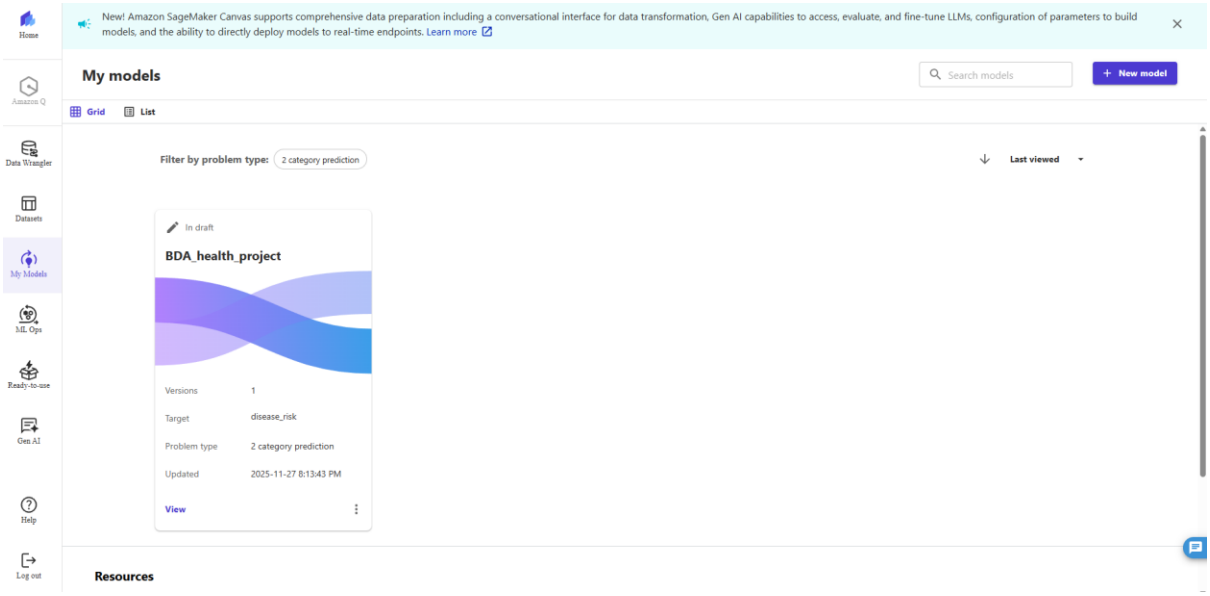
3. Canvas Dashboard



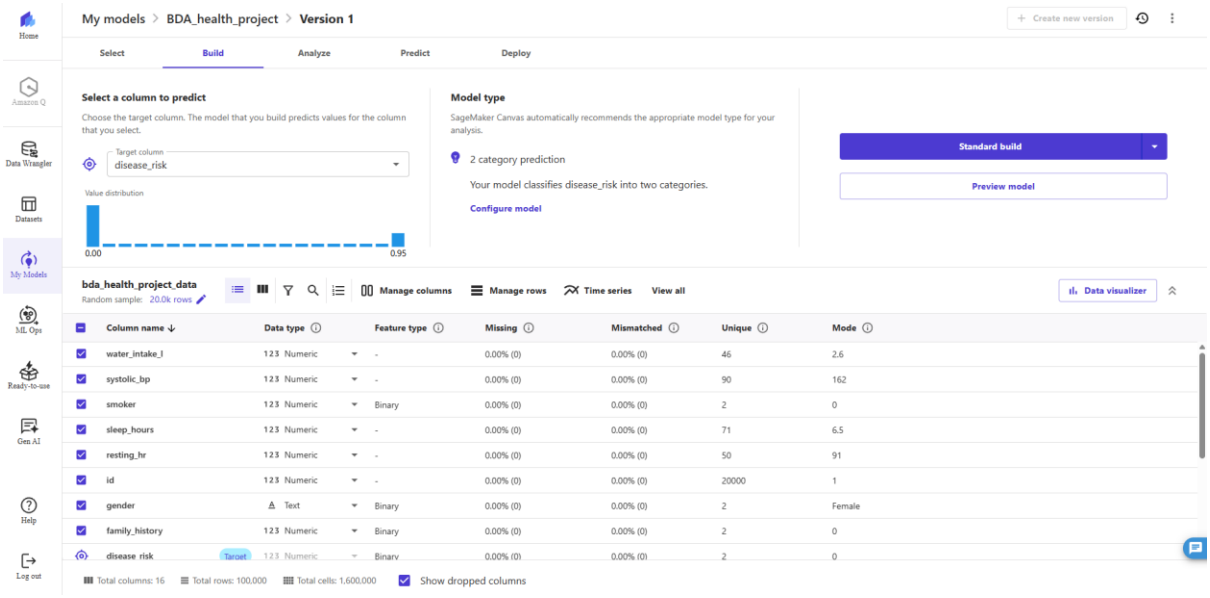
4. Uploaded data from S3



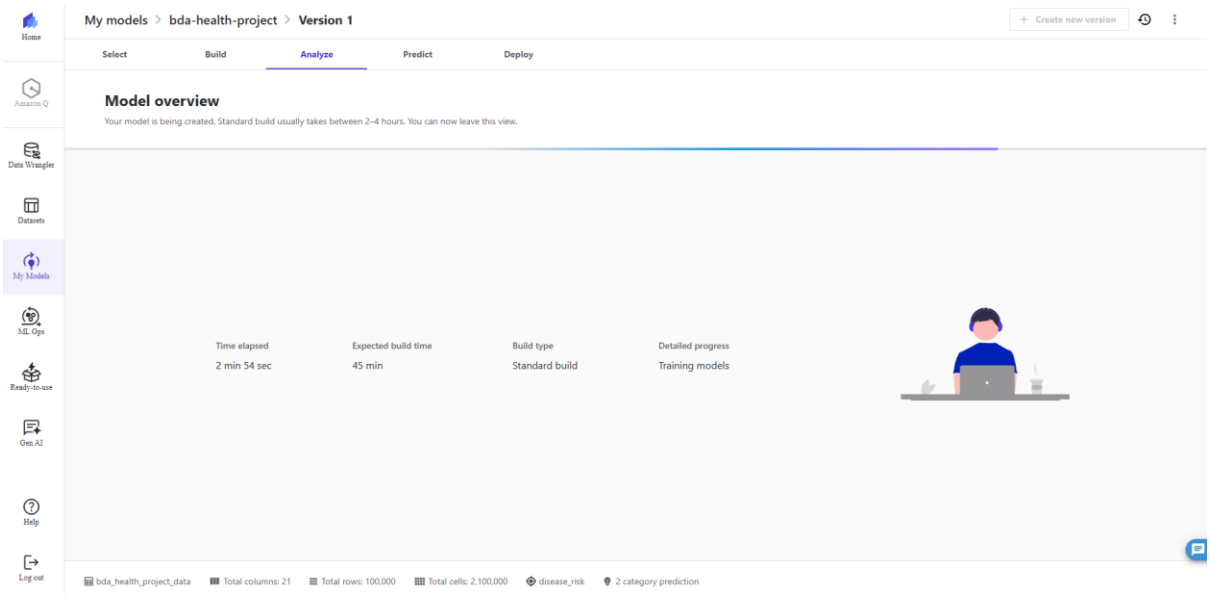
5. Creating a new model in canvas autopilot



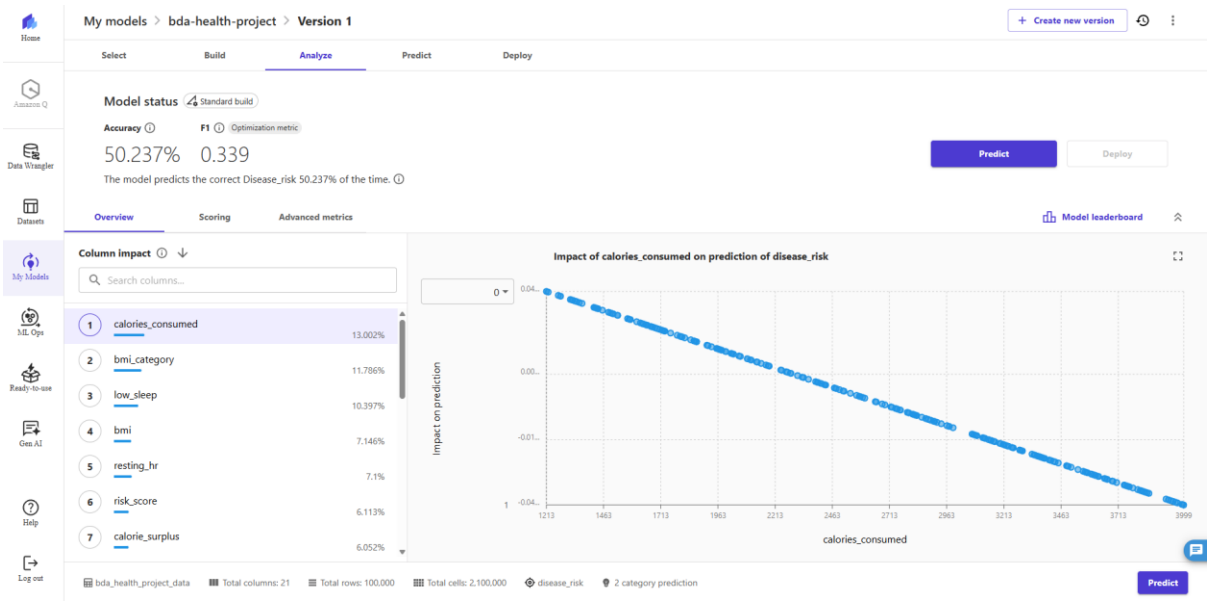
6. Selecting target column disease_risk



7. Building Model



8. Analysing model



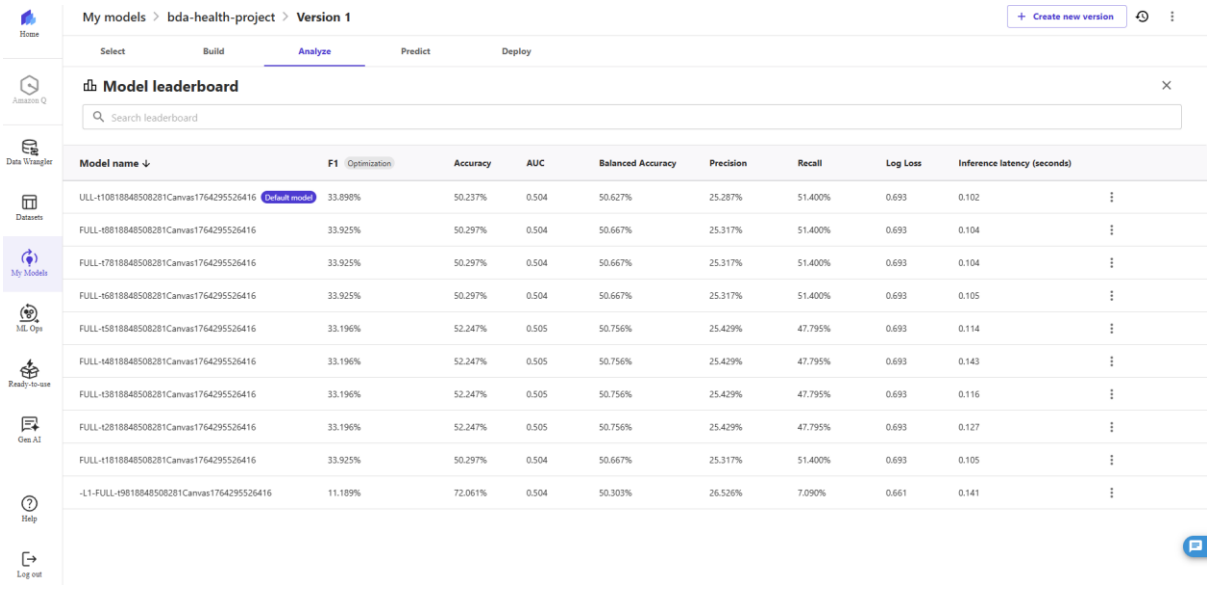
Model Performance Summary

The SageMaker Autopilot experiment completed successfully using a Standard Build. The best model achieved:

- **Accuracy: 50.23 percent**
- **F1 Score: 0.339**

This indicates **moderate predictive performance** on the disease_risk classification task, suggesting the dataset may need additional feature engineering or balancing.

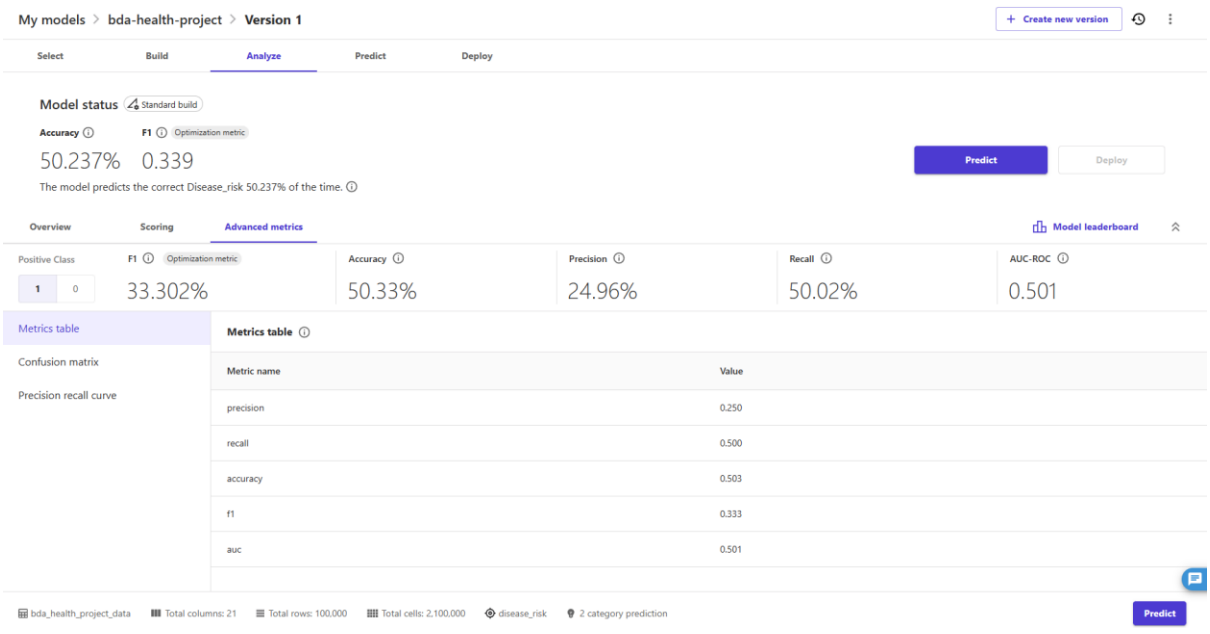
9. Model Leaderboard



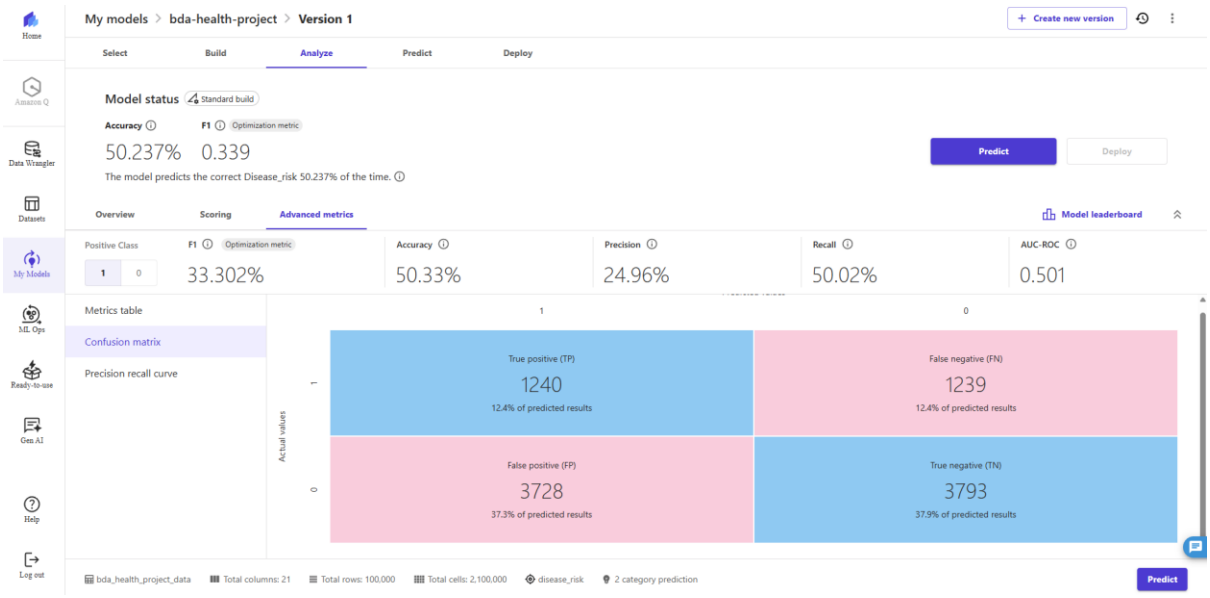
Model Leaderboard Findings

Autopilot evaluated multiple algorithms and displayed them on the Model Leaderboard. The best-performing model was automatically selected based on the optimization metric (F1 score). The leaderboard provided a comparison of candidate models along with their key metrics, allowing quick identification of the most effective model for this classification task.

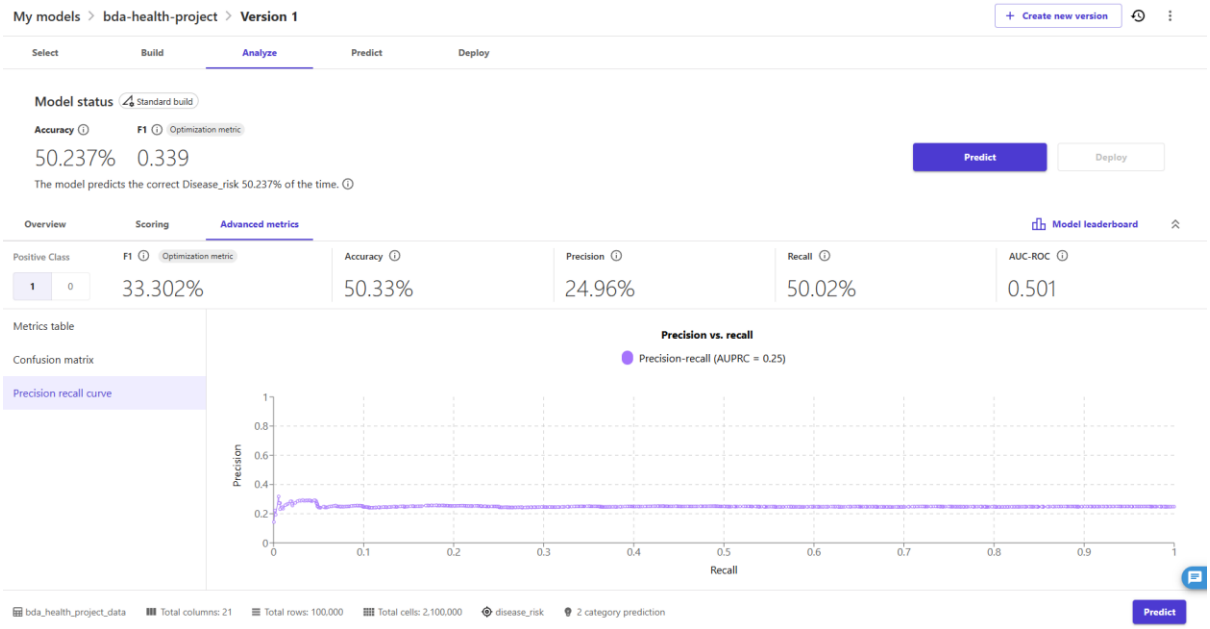
10. Matrix Table



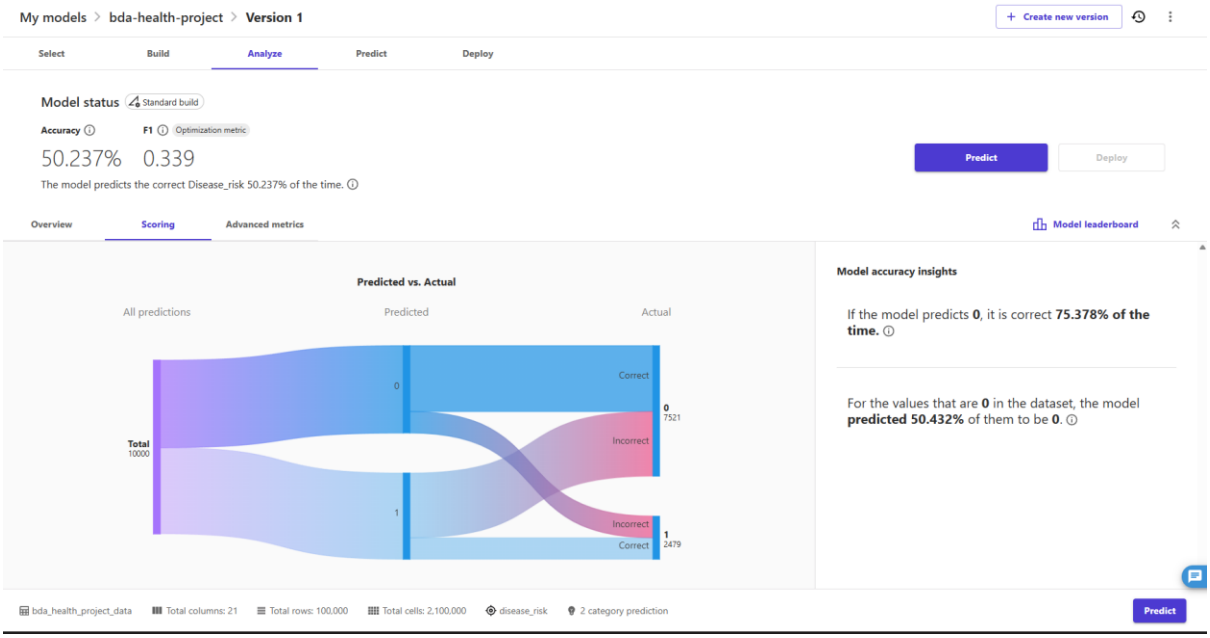
11. Confusion Matrix



12. Precision Recall



13. Scoring



Advanced Evaluation Metrics

The confusion matrix and ROC curve helped assess the model's classification quality. The confusion matrix revealed the distribution of correct and incorrect predictions, highlighting areas where the model struggled to differentiate between risk classes. The ROC curve indicated the trade-off between true positive and false positive rates, offering additional insights into overall model performance beyond simple accuracy.

Ethical Considerations

A. Bias in Training Data

- The dataset includes sensitive attributes like **gender**, **family_history**, **smoker**, and **sleep patterns**, which may introduce bias.
- If these categories are **not evenly distributed**, the model may give unfair predictions for underrepresented groups.
- Strongly correlated features (such as BMI or smoking status) can cause the model to **over-emphasize certain lifestyle habits**, reducing fairness.
- The model accuracy and F1 score indicate that additional checks (class balancing, feature selection) may be needed to **prevent biased outcomes**.

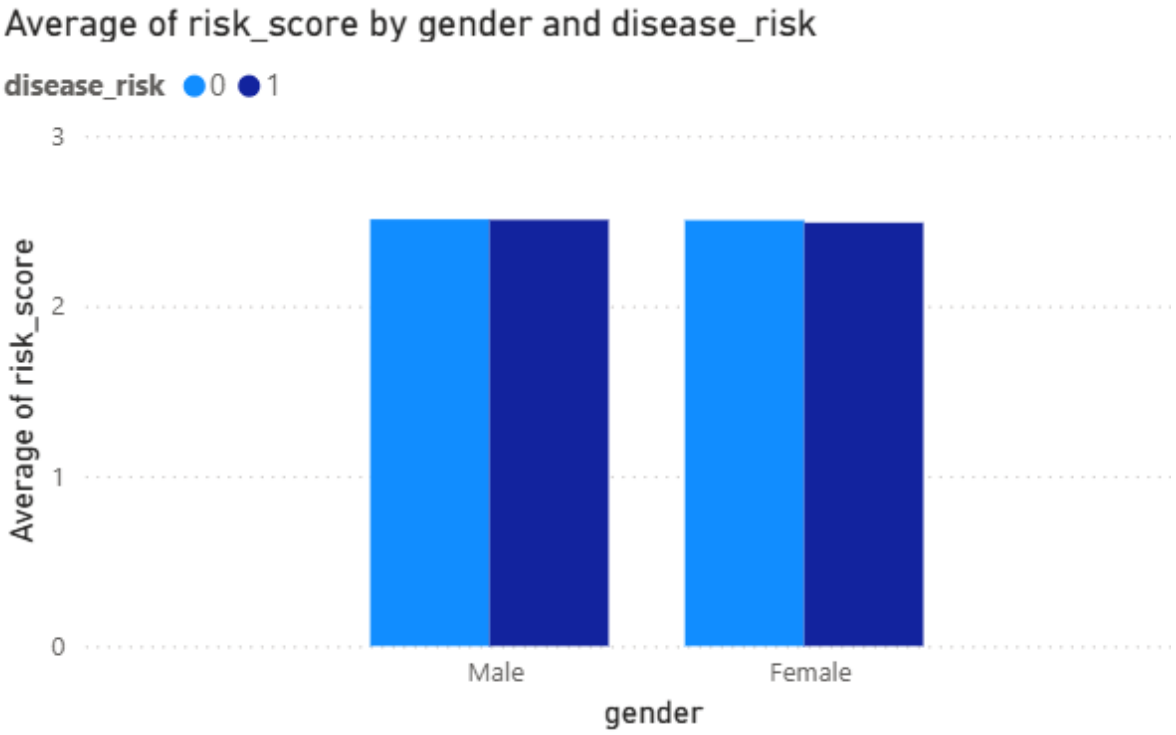
B. Privacy Concerns

- Health and lifestyle data is sensitive and must be stored securely in **Amazon S3 with encryption**.
- Access should be restricted using **IAM least-privilege permissions** to avoid unauthorized viewing of personal data.
- Identifiable fields (like **ID**) should be removed or anonymized to protect individuals' privacy.
- All processing in SageMaker should follow standard **health data privacy practices**, ensuring no personal identity or confidential information is exposed.

Visualization

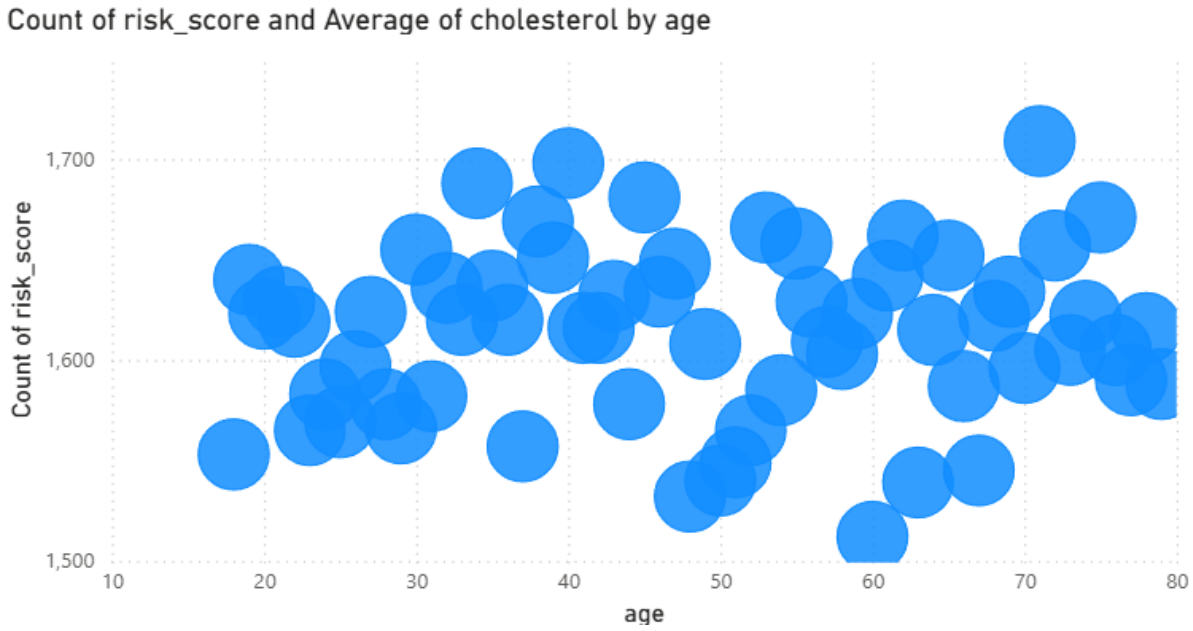
Visualization 1: Average of risk_score by gender and disease_risk
Chart Type: Clustered Column Chart
Description: This visualization displays the average risk score segmented by gender (Male and Female) and disease risk status (0 = No disease risk, 1 = Has disease risk). The chart shows two bars for each gender - a light blue bar representing individuals without disease risk (0) and a dark blue bar representing those with disease risk (1). Both males and females show similar average risk scores of approximately 2.5 across both disease risk categories, indicating consistent risk distribution regardless of gender.
Key Insights:

- Males and females have nearly identical average risk scores
- Disease risk status shows minimal variation in risk scores between the two groups
- Both genders show average risk scores between 2-3 on the scale



Visualization 2: Count of risk_score and Average of cholesterol by age
Chart Type: Scatter Plot (Bubble Chart)
Description: This scatter plot illustrates the relationship between age (X-axis), count of risk scores (Y-axis represented by bubble size), and average cholesterol levels. Each blue bubble represents an age group, with the size indicating the count of individuals at that age. The visualization spans ages from approximately 10 to 80 years, with count values ranging from 1,500 to 1,700. The bubble distribution shows relatively even representation across all age groups.
Key Insights:

- Sample size is fairly consistent across all age groups
- Age distribution is well-balanced throughout the dataset
- Most age groups have between 1,550-1,700 records
- Cholesterol levels can be analyzed in relation to age through bubble positioning

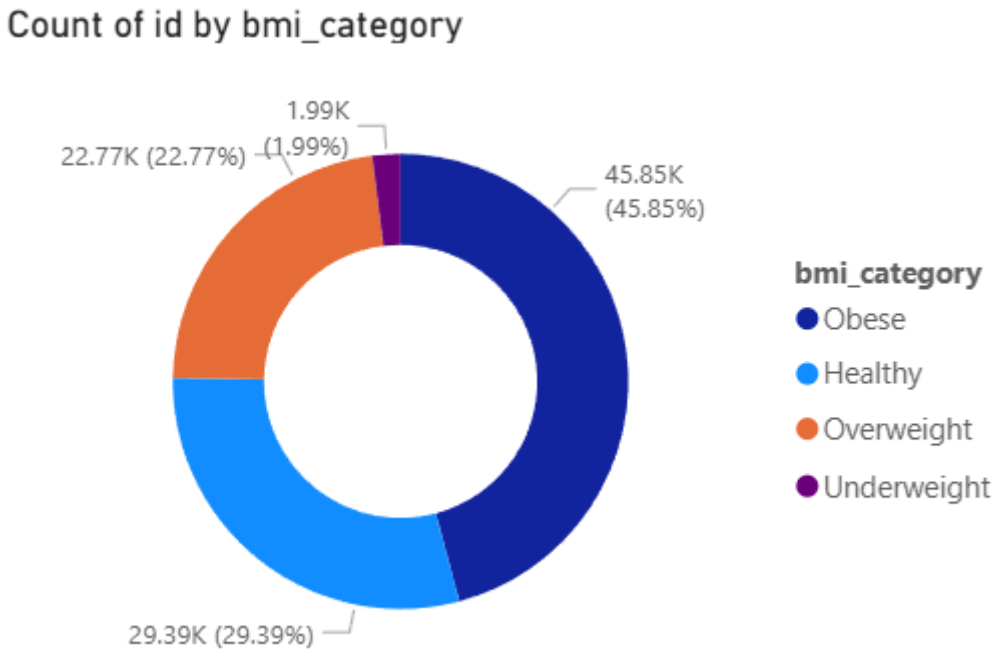


Visualization 3: Count of id by bmi_category
Chart Type: Donut Chart
Description: This donut chart presents the distribution of the population across four BMI categories: Obese (dark blue), Healthy (light blue), Overweight (orange), and Underweight (purple). The chart shows that:

- **Obese:** 45,854 individuals (45.85%)
- **Healthy:** 29,391 individuals (29.39%)
- **Overweight:** 22,777 individuals (22.77%)
- **Underweight:** 1,990 individuals (1.99%)

Key Insights:

- Nearly half (45.85%) of the population falls into the obese category
- About one-third (29.39%) maintain a healthy BMI
- Underweight individuals represent less than 2% of the population
- Combined overweight and obese categories represent 68.62% of the population



Visualization 4: BMI Category Matrix by Disease Risk
Chart Type: Matrix/Table
Description: This matrix visualization cross-tabulates BMI categories with disease risk status (0 = No disease risk, 1 = Has disease risk). The table displays:

- **Healthy:** 8,135 without disease risk, 21,256 with disease risk (Total: 29,391)
- **Obese:** 12,792 without disease risk, 33,062 with disease risk (Total: 45,854)
- **Overweight:** 6,250 without disease risk, 16,515 with disease risk (Total: 22,765)
- **Underweight:** 559 without disease risk, 1,431 with disease risk (Total: 1,990)
- **Overall Total:** 27,736 without disease risk, 72,264 with disease risk (100,000 total)

Key Insights:

- 72.26% of the population has disease risk (72,264 out of 100,000)
- Obese individuals have the highest disease risk count (33,062)
- Even healthy BMI individuals show significant disease risk (21,256)
- Disease risk is present across all BMI categories

bmi_category	0	1	**Total**
Healthy	8135	21256	**29391**
Obese	12792	33062	**45854**
Overweight	6250	16515	**22765**
Underweight	559	1431	**1990**
Total	**27736**	**72264**	**100000**

Visualization 5: Key Performance Indicator Cards
Chart Type: KPI Cards (4 cards)
Card 1 - Average of BMI: 29.02

- Displays the overall average BMI across the entire population
- A BMI of 29.02 falls into the "Overweight" category (25-29.9)
- Indicates the population generally trends toward higher BMI values

Card 2 - Average of daily_steps: 10.48K

- Shows the average daily step count is 10,480 steps
- Exceeds the commonly recommended 10,000 steps per day
- Suggests moderate to good physical activity levels in the population

Card 3 - Average of sleep_hours: 6.49

- Displays average sleep duration of 6.49 hours per night
- Falls below the recommended 7-9 hours of sleep for adults
- May indicate potential sleep deficiency across the population

Card 4 - Average of risk_score: 2.50

- Shows overall average health risk score of 2.50
- On a scale where higher values indicate greater risk
- Represents a moderate risk level across the population

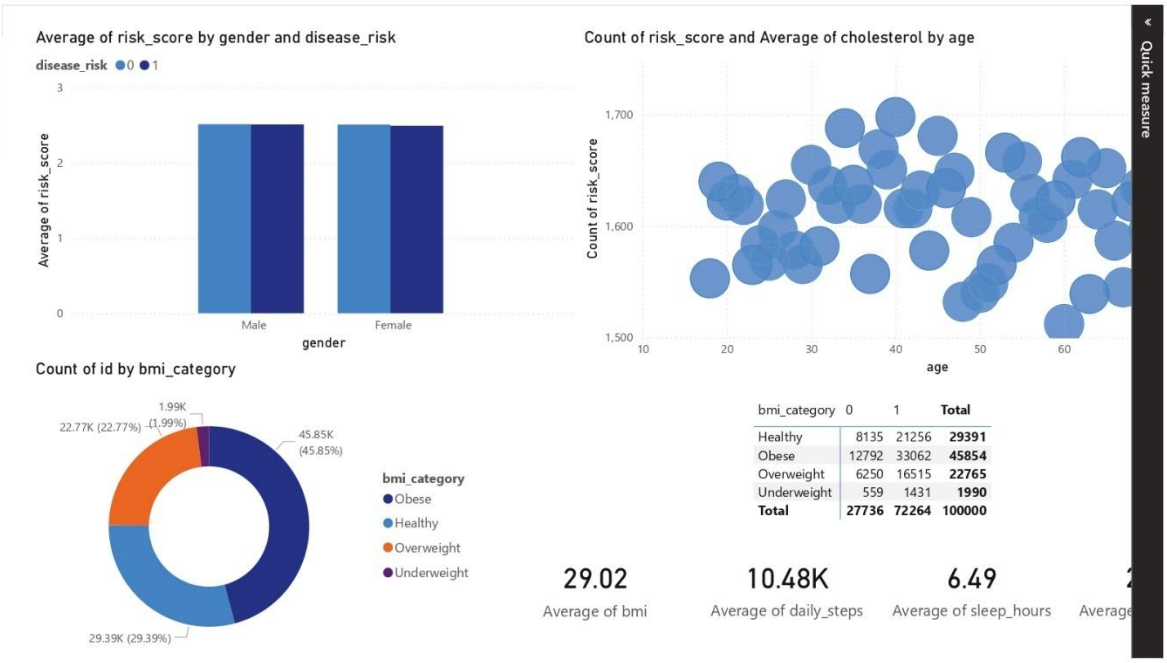
Combined Insights from KPI Cards:

- Despite adequate physical activity (10,480 steps), the population shows elevated BMI (29.02)
- Insufficient sleep (6.49 hours) may contribute to overall health risk
- The moderate risk score (2.50) aligns with the high percentage of individuals with disease risk

Overall Dashboard Summary
Dashboard Title: Health Risk Analysis Dashboard
Purpose: This interactive dashboard provides comprehensive insights into the health status and risk factors of a population of 100,000 individuals, analyzing the relationships between demographics, lifestyle factors, BMI categories, and disease risk.
Key Findings:

1. The population shows significant health concerns with 72% having disease risk
2. BMI distribution skews heavily toward overweight/obese categories (68.6%)
3. Physical activity levels are adequate, but sleep duration is below recommended levels
4. Disease risk is distributed across all BMI categories, not just obese individuals
5. Gender shows minimal impact on risk scores
6. Age distribution is balanced, allowing for fair comparison across age groups

Data Source: Combined health dataset (combine.csv) containing 100,000 records with 21 health and lifestyle variables



Technical Notes

Tools Used: Microsoft Power BI Desktop

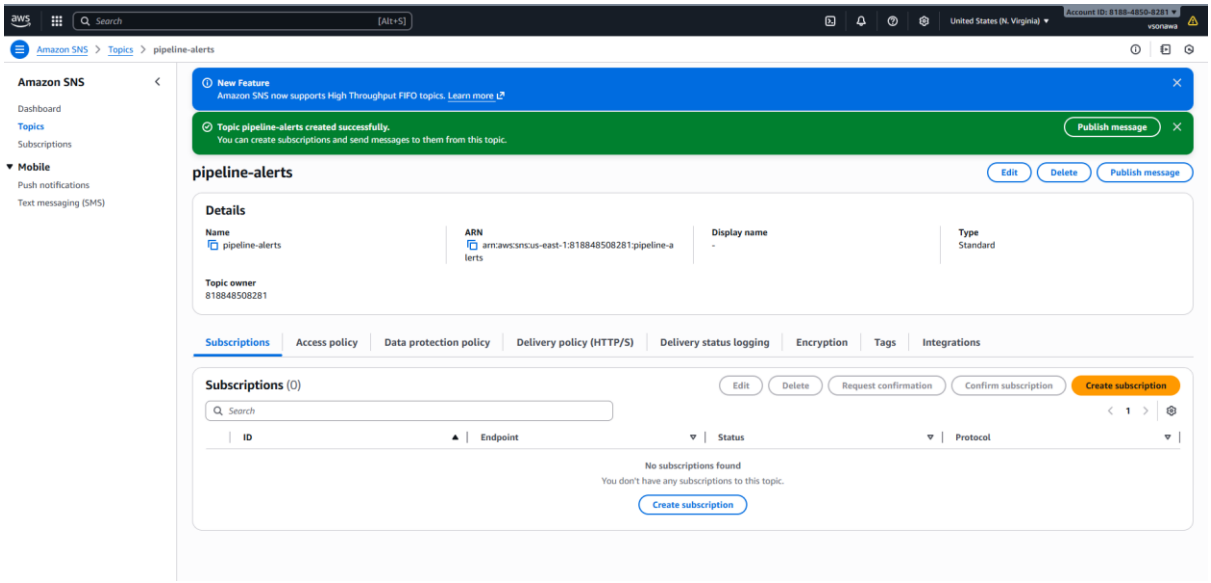
Data Processing:

- Dataset imported from CSV format
- No data transformation required (data was pre-processed in PySpark)
- Total records: 100,000
- Total fields analyzed: 21 variables

Visualizations Created: 5 main visualizations plus 4 KPI cards

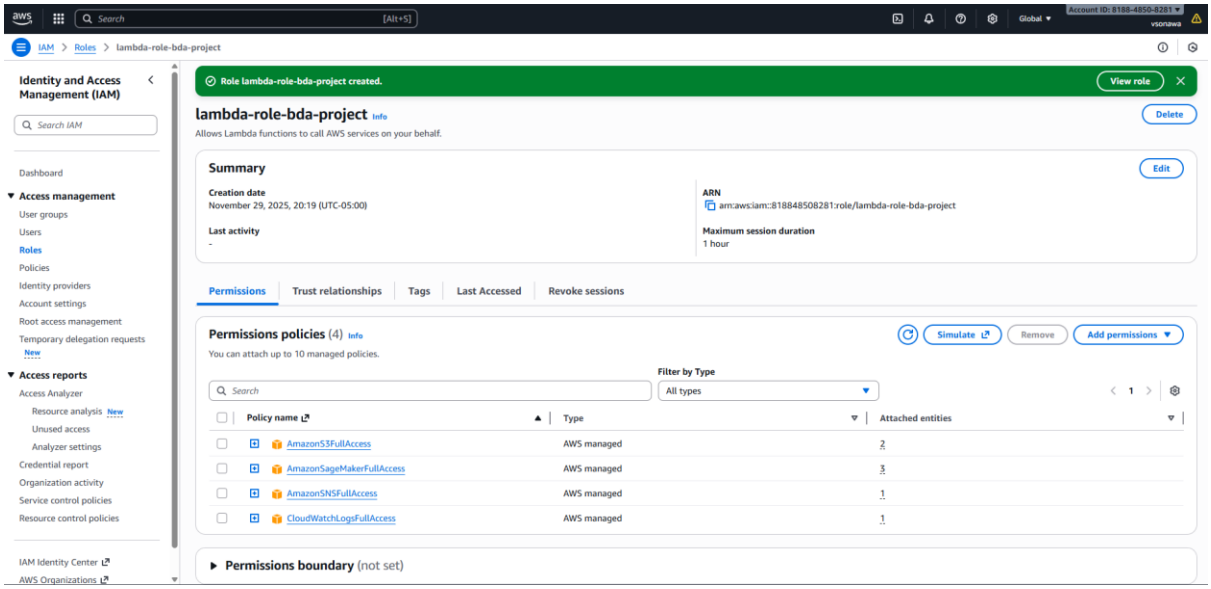
Automation

Creating an SNS topic pipeline-alerts and subscription to email in AWS Console

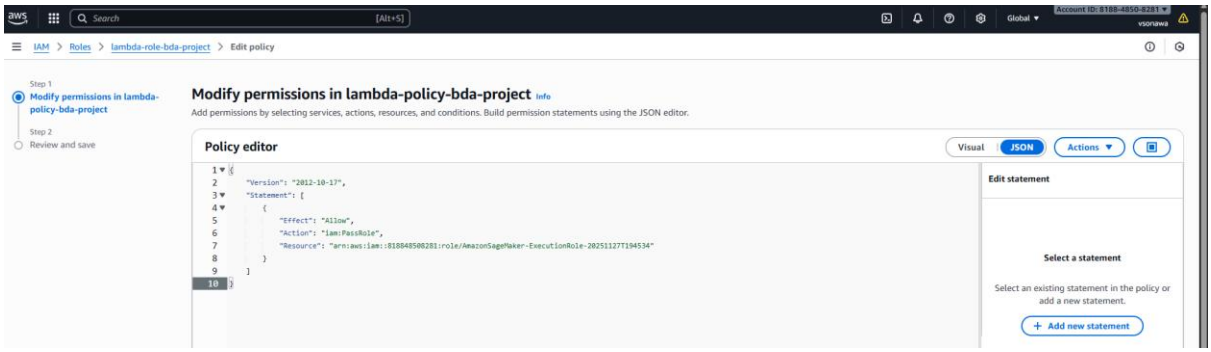


Creating IAM role for AWS Lambda

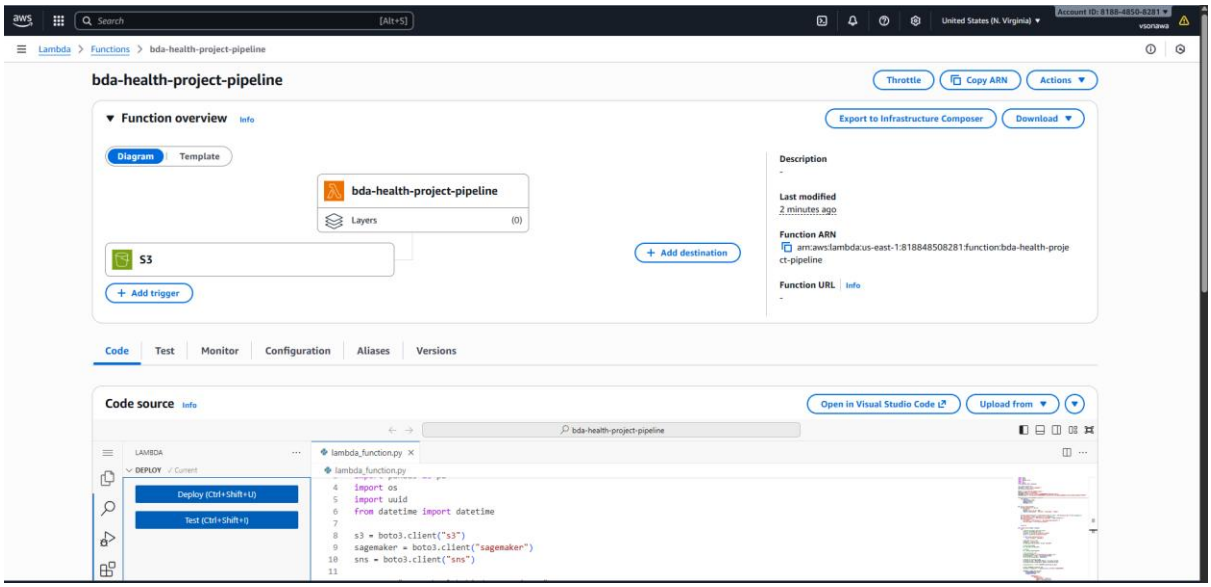
- **AmazonS3FullAccess**
  (or AmazonS3ReadOnlyAccess + one custom PutObject policy)
- **AmazonSageMakerFullAccess**
  (this includes CreateAutoMLJob)
- **AmazonSNSFullAccess**
  (for publishing messages)
- **CloudWatchLogsFullAccess**
  (Lambda log permissions)



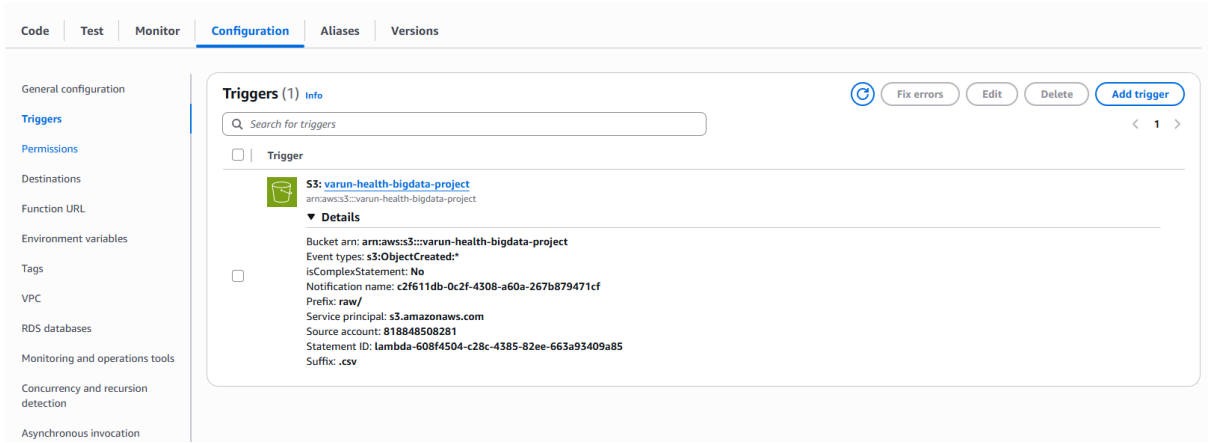
Also created lambda policy to pass the SageMaker execution role to Autopilot.



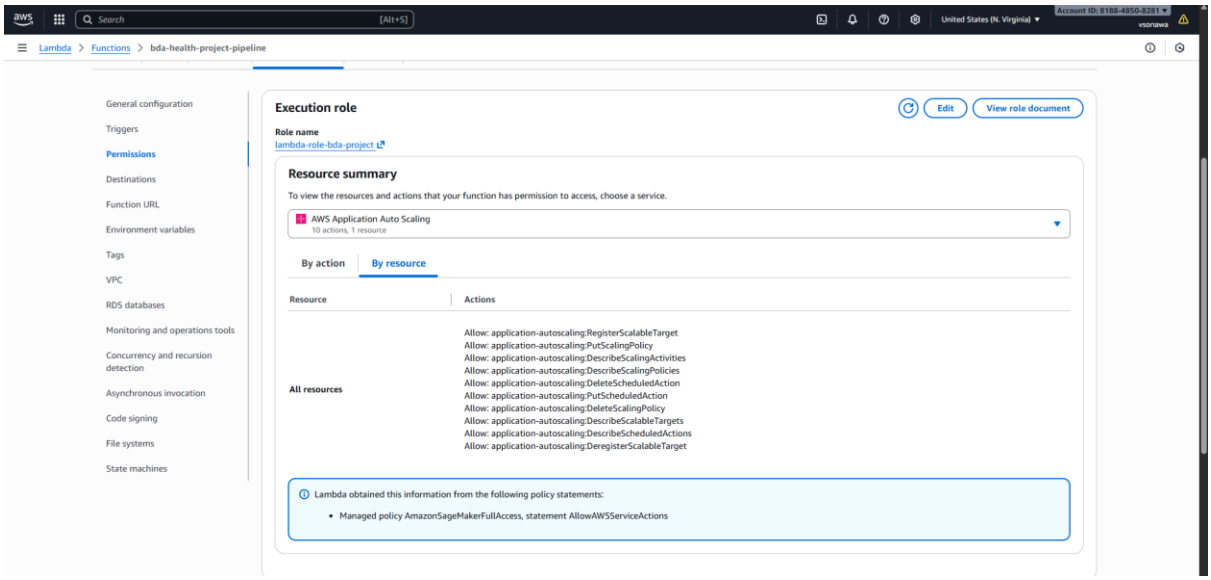
Created a Lambda Function



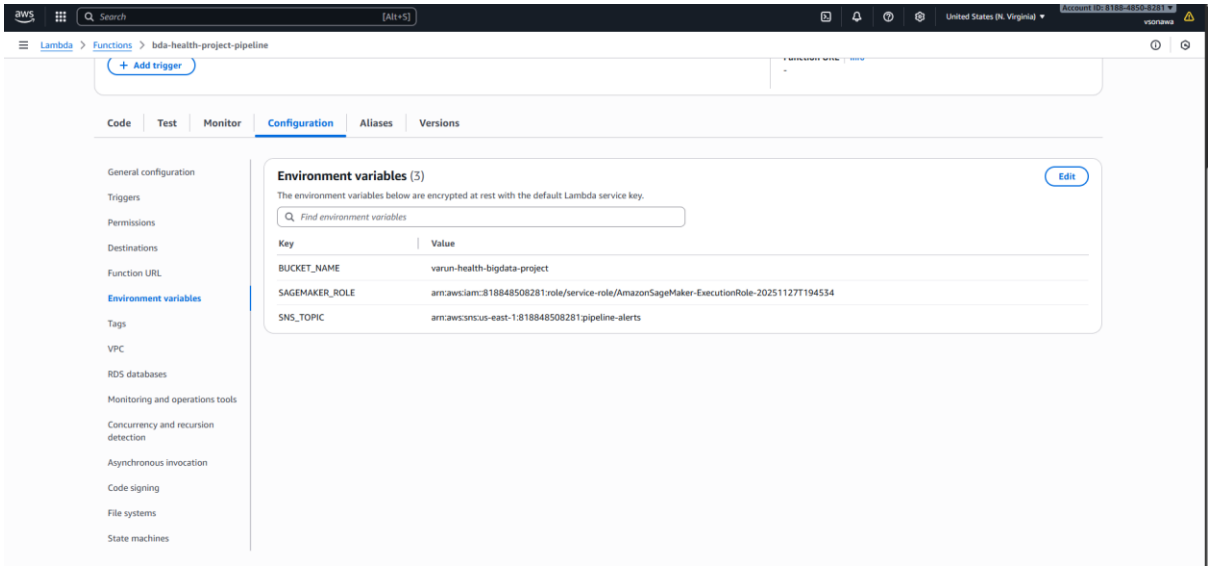
Added a new Trigger for s3 (This allows code to read .csv file from S3 bucked /raw folder)



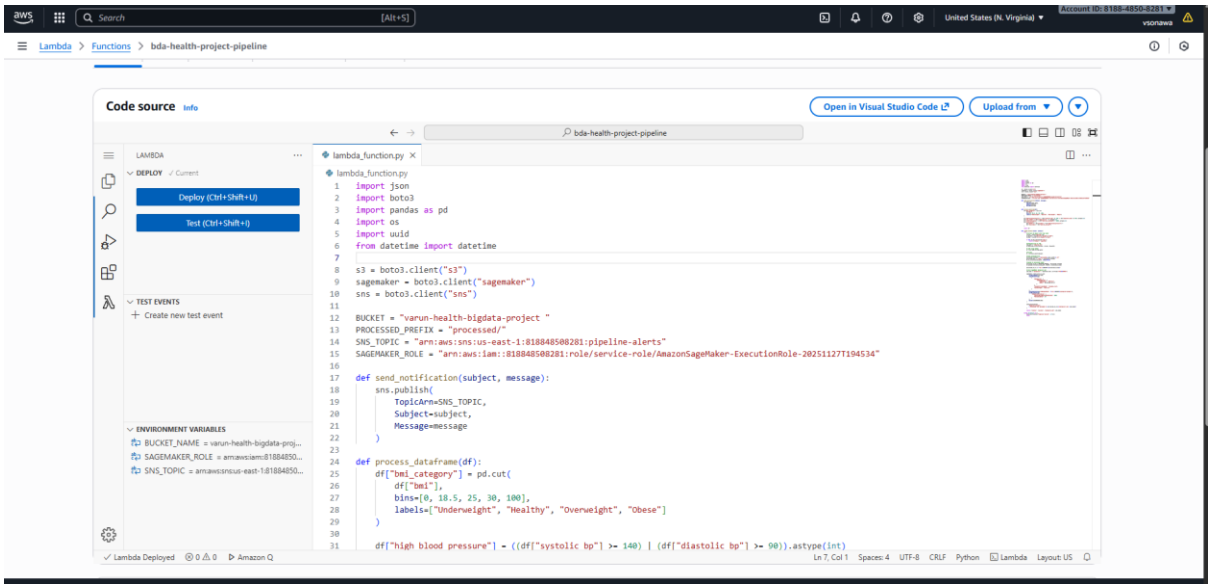
Attached the new created IAM role for AWS Lambda



Added Environment variables



Wrote and Deployed the Python Script



What the Lambda script does

1. Gets triggered when a CSV is uploaded to raw/ in S3.
2. Downloads the raw CSV into Lambda.
3. Loads the data using Pandas.(instead of pyspark because for that I need to use AWS Glue instead of Lambda and it will cost a lot.)
4. Applies all preprocessing steps and creates new columns.
5. Saves the processed CSV to processed/ in S3.
6. Starts a SageMaker Autopilot job using the processed file.
7. Sends an SNS notification on success.
8. Sends an SNS notification on failure.

Lambda.function.py

```
import json
import boto3
import pandas as pd
import os
import uuid
from datetime import datetime

s3 = boto3.client("s3")
sagemaker = boto3.client("sagemaker")
sns = boto3.client("sns")

BUCKET = "varun-health-bigdata-project "
PROCESSED_PREFIX = "processed/"
SNS_TOPIC = "arn:aws:sns:us-east-1:818848508281:pipeline-alerts"
SAGEMAKER_ROLE = "arn:aws:iam::818848508281:role/service-role/AmazonSageMaker-ExecutionRole-20251127T194534"
```

```
def send_notification(subject, message):
    sns.publish(
        TopicArn=SNS_TOPIC,
        Subject=subject,
        Message=message
    )

def process_dataframe(df):
    df["bmi_category"] = pd.cut(
        df["bmi"],
        bins=[0, 18.5, 25, 30, 100],
        labels=["Underweight", "Healthy", "Overweight", "Obese"]
    )

    df["high_blood_pressure"] = ((df["systolic_bp"] >= 140) | (df["diastolic_bp"] >=
90)).astype(int)
    df["low_sleep"] = (df["sleep_hours"] < 7).astype(int)
    df["calorie_surplus"] = (df["calories_consumed"] > 2000).astype(int)
    df["risk_score"] = (
        df["smoker"] + df["alcohol"] + df["high_blood_pressure"] +
        df["low_sleep"] + df["calorie_surplus"]
    )

    return df

def lambda_handler(event, context):
    try:
        # Extract S3 object info from event
        record = event["Records"][0]
        s3_bucket = record["s3"]["bucket"]["name"]
        s3_key = record["s3"]["object"]["key"]

        if not s3_key.startswith("raw/"):
            return {"status": "ignored"}

        # Download file to /tmp
        tmp_path = "/tmp/raw.csv"
        s3.download_file(s3_bucket, s3_key, tmp_path)

        # Load using pandas
        df = pd.read_csv(tmp_path)

        # Process
        df = process_dataframe(df)

        # Save processed file
```

```
    processed_filename = f"processed_{uuid.uuid4()}.csv"
    processed_path = "/tmp/" + processed_filename
    df.to_csv(processed_path, index=False)

    # Upload to processed folder
    processed_s3_key = PROCESSED_PREFIX + processed_filename
    s3.upload_file(processed_path, BUCKET, processed_s3_key)

    processed_s3_uri = f"s3://{BUCKET}/{processed_s3_key}"

    # Start SageMaker Autopilot job
    job_name = "autopilot-" + datetime.now().strftime("%Y%m%d%H%M%S")

    sagemaker.create_auto_ml_job(
      AutoMLJobName=job_name,
      InputDataConfig=[
        {
          "DataSource": {
            "S3DataSource": {
              "S3DataType": "S3Prefix",
              "S3Uri": processed_s3_uri
            }
          },
          "TargetAttributeName": "disease_risk",
          "ContentType": "text/csv"
        }
      ],
      OutputDataConfig={"S3OutputPath": f"s3://{BUCKET}/autopilot-output/"},
      AutoMLJobConfig={
        "CompletionCriteria": {
          "MaxAutoMLJobRuntimeInSeconds": 3600,
          "MaxCandidates": 10
        }
      },
      RoleArn=SAGEMAKER_ROLE
    )

    send_notification(
      "Pipeline success",
      f"Processed file uploaded to {processed_s3_uri}\nAutopilot job: {job_name}"
    )

    return {"status": "success", "autopilot_job": job_name}

  except Exception as e:
    send_notification("Pipeline failure", str(e))
    raise
```

PowerBI Automation:

The visualization component is not automated because the my project uses PowerBI Desktop instead of AWS QuickSight. PowerBI Desktop is an offline tool, and it does not support automatic refresh through serverless workflows. Based on the professor's permission, the dashboard visualizations created in PowerBI have been included directly in the report, while the rest of the data pipeline is fully automated through AWS services.

Conclusions

This project successfully implemented an end to end big data pipeline using AWS and PySpark. Large scale health data was ingested from S3, processed in a distributed PySpark environment, and analyzed through feature engineering, aggregations, and SQL queries. AWS SageMaker Autopilot automated the machine learning process, providing model evaluations and highlighting areas for improvement such as feature enhancement and class balance.

The serverless automation pipeline using S3 triggers, Lambda, SageMaker, and SNS demonstrated how new data can be processed and modeled without manual steps, reflecting real world data engineering workflows. Power BI visualizations effectively summarized health patterns, BMI distribution, lifestyle risks, and disease likelihood across the population.

Overall, the project achieved its goals by integrating scalable processing, automated ML, and interactive visualization into a unified cloud based analytics solution.

References

- Amazon Web Services. “Amazon S3 Documentation.” AWS, https://docs.aws.amazon.com/s3/.
- Amazon Web Services. “Amazon EC2 Documentation.” AWS, https://docs.aws.amazon.com/ec2/.
- Amazon Web Services. “AWS Lambda Documentation.” AWS, https://docs.aws.amazon.com/lambda/.
- Amazon Web Services. “Amazon SageMaker Autopilot Documentation.” AWS, https://docs.aws.amazon.com/sagemaker/latest/dg/autopilot-automate-model-development.html.
- Apache Software Foundation. “PySpark Documentation.” https://spark.apache.org/docs/latest/api/python/.
- Microsoft Corporation. “Power BI Desktop Documentation.” https://learn.microsoft.com/power-bi/.
- Kaggle. “Dataset Source.” https://www.kaggle.com/.
- OpenAI. “ChatGPT.” Used as a support tool for understanding AWS setup, exploring debugging approaches, and clarifying implementation steps.