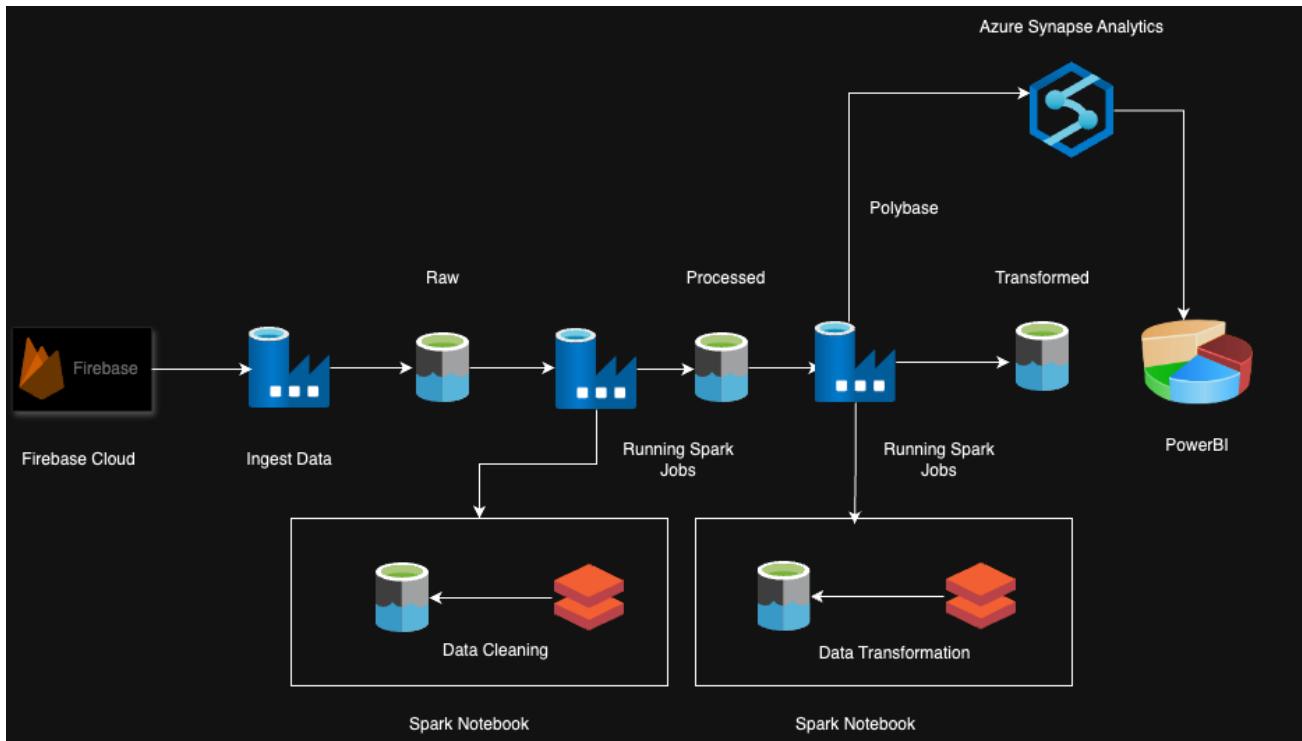


Shreehari

Meet Patel

PROJECT – HEALTH DATA ANALYSIS

Architecture Diagram

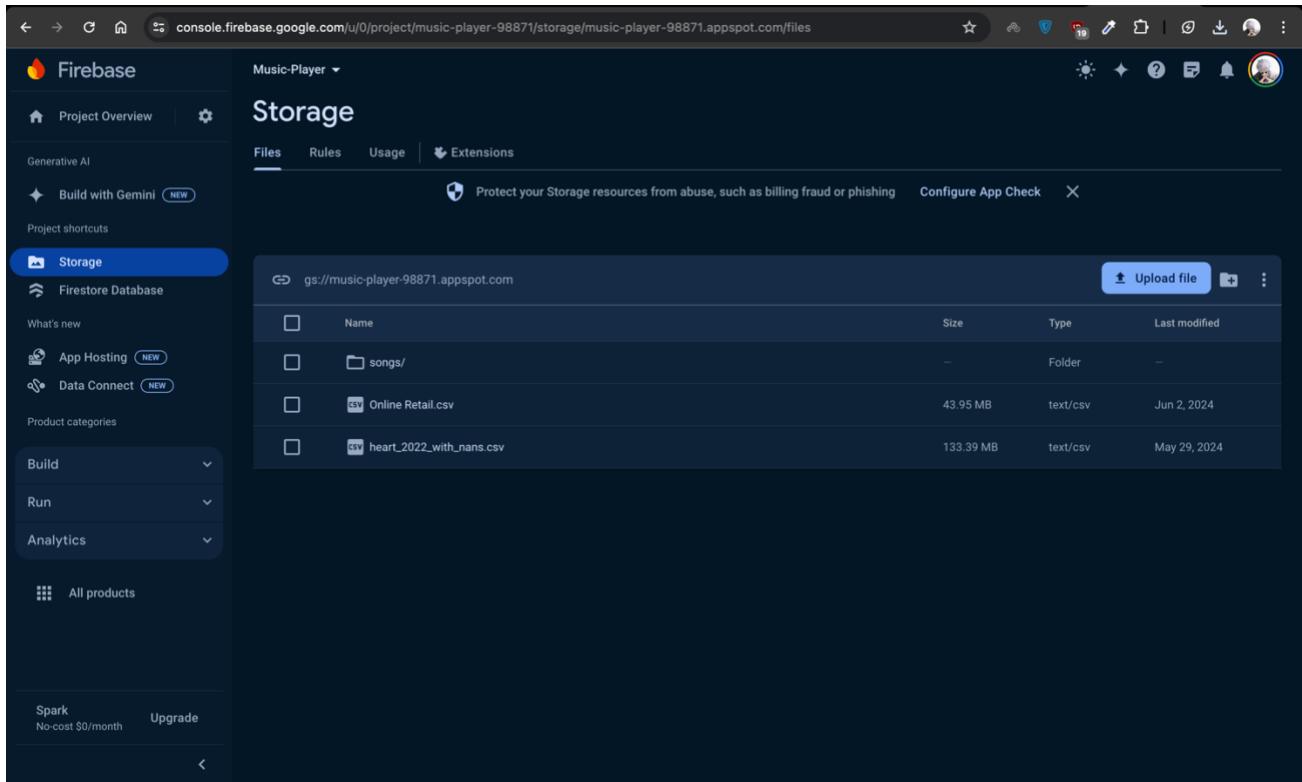


Raw Data of Health Dataset

Possible Data Loss: Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	
1	State	Sex	GeneralHealth	PhysicalHealth	MentalHealth	LastCheckup	PhysicalActivity	SleepHours	RemovedTee	HadHeartAttack	HadAngina	HadStroke	HadAsthma	HadSkinCancer	HadCOPD	HadDepression	HadKidneyDisease	HadArthritis	HadDiabetes	DeafOrHardOfHearing	BlindOrVisionImpaired	DroppedOut
2	Alabama	Female	Very good	0	0	Within past year	No	8	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No
3	Alabama	Female	Excellent	0	0	Within past year	No	6	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No
4	Alabama	Female	Very good	2	3	Within past year	Yes	5	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No
5	Alabama	Female	Excellent	0	0	Within past year	Yes	7	No	No	No	Yes	No	No	No	No	No	Yes	No	No	No	No
6	Alabama	Female	Fair	2	0	Within past year	Yes	9	No	No	No	No	No	No	No	No	No	No	No	No	No	No
7	Alabama	Male	Poor	1	0	Within past year	No	7	Yes	No	Yes	No	No	No	No	No	No	No	Yes	No	No	No
8	Alabama	Female	Very good	0	0	Within past year	Yes	7	No	No	No	No	No	No	No	No	No	No	No	No	No	No
9	Alabama	Female	Good	0	0	Within past year	No	8	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No
10	Alabama	Female	Good	0	0	Within past year	Yes	6	No	No	No	Yes	No	No	No	No	No	Yes	No	No	Yes	No
11	Alabama	Female	Good	1	0	Within past year	Yes	7	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No
12	Alabama	Female	Fair	8	9	Within past year	No	8	No	No	No	No	No	No	No	No	No	No	No	No	No	No
13	Alabama	Female	Good	0	0	Within past year	No	6	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No
14	Alabama	Male	Fair	5	0	Within past year	No	6	No	No	Yes	No	No	No	No	Yes	No	No	No	No	No	No
15	Alabama	Male	Very good	0	0	Within past year	Yes	8	No	No	No	No	No	No	No	No	No	No	No	No	No	No
16	Alabama	Female	Good	30	5	Within past year	Yes	8	No	No	No	Yes	No	No	No	Yes	No	Yes	No	Yes	No	No
17	Alabama	Female	Excellent	0	0	Within past year	Yes	8	No	No	No	No	No	No	No	No	No	No	No	No	No	No
18	Alabama	Female	Excellent	0	0	Within past year	Yes	6	No	No	No	No	No	No	No	No	No	No	No	No	No	No
19	Alabama	Female	Fair	0	15	Within past year	Yes	6	No	No	No	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No	No
20	Alabama	Female	Poor	0	0	Within past year	Yes	4	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No
21	Alabama	Female	Very good	0	0	Within past year	Yes	6	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No
22	Alabama	Female	Very good	4	5	Within past year	Yes	6	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No	No
23	Alabama	Male	Good	0	0	Within past year	Yes	9	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No
24	Alabama	Female	Fair	30	0	Within past year	Yes	7	No	Yes	No	No	Yes	No	No	No	Yes	No	No	No	No	No
25	Alabama	Female	Poor	30	0	Within past year	No	8	No	No	No	Yes	No	No	No	Yes	Yes	No	No	No	Yes	No
26	Alabama	Male	Excellent	0	0	Within past year	Yes	8	No	No	No	Yes	No	No	No	Yes	No	No	No	No	No	No
27	Alabama	Female	Very good	0	0	Within past year	Yes	6	No	No	No	Yes	No	No	No	Yes	No	No	No	No	No	No
28	Alabama	Female	Fair	23	3	Within past year	Yes	8	Yes	Yes	No	No	No	No	No	Yes	Yes	No	No	No	No	No
29	Alabama	Female	Very good	0	3	Within past year	No	8	No	No	No	No	No	No	No	Yes	No	Yes	No	Yes	No	No
30	Alabama	Female	Good	0	20	Within past year	No	8	No	Yes	No	No	Yes	No	No	Yes	No	No	No	No	No	No
31	Alabama	Female	Good	0	0	Within past year	No	8	No	No	No	No	No	No	No	No	Yes	Yes	Yes	No	No	No
32	Alabama	Female	Good	2	3	Within past year	Yes	7	No	No	No	No	No	No	No	No	Yes	No	Yes	No	No	No
33	Alabama	Male	Fair	30	0	Within past year	Yes	4	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	No	No	No
34	Alabama	Female	Poor	14	14	Within past year	No	6	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes
35	Alabama	Male	Very good	0	0	Within past year	Yes	6	No	No	No	No	No	No	No	No	No	No	No	No	No	No
36	Alabama	Female	Fair	0	0	Within past year	Yes	10	Yes	Yes	No	No	No	No	No	Yes	No	No	No	No	No	No

Data in Firebase cloud



The screenshot shows the Firebase Storage interface for the project "Music-Player". The left sidebar includes links for Project Overview, Generative AI, Build with Gemini, Project shortcuts, Storage (which is selected and highlighted in blue), Firestore Database, App Hosting, Data Connect, Product categories, Build, Run, Analytics, and All products. The main area displays a list of files under the path `gs://music-player-98871.appspot.com`. The files listed are:

Name	Type	Last modified
<code>songs/</code>	Folder	-
<code>Online Retail.csv</code>	text/csv	Jun 2, 2024
<code>heart_2022_with_nans.csv</code>	text/csv	May 29, 2024

At the top right of the main area, there are buttons for "Upload file" and "Extensions". The top navigation bar shows the URL `console.firebaseio.google.com/u/0/project/music-player-98871/storage/music-player-98871.appspot.com/files`.

Git Repo

The screenshot shows the GitHub interface for creating a new repository. At the top, there's a navigation bar with icons for search, issues, pull requests, and user profile. Below it, a header says "Create a new repository". A sub-header notes that a repository contains all project files, including revision history, and provides a link to "Import a repository". A note states that required fields are marked with an asterisk (*).

The main form fields include:

- Owner ***: meet74
- Repository name ***: project_1
- Description (optional)**: (empty text area)
- Visibility**:
 - Public**: Anyone on the internet can see this repository. You choose who can commit.
 - Private**: You choose who can see and commit to this repository.
- Initialize this repository with:**
 - Add a README file**: This is where you can write a long description for your project. [Learn more about READMEs](#).
- Add .gitignore**: .gitignore template: None
- Choose a license**: License: None
- Note**: A license tells others what they can and can't do with your code. [Learn more about licenses](#). This will set `main` as the default branch. Change the default name in your settings.
- Information**: You are creating a public repository in your personal account.

At the bottom right is a green "Create repository" button.

Azure Data Lake Storage and Azure Data Factory

Creating Azure Data Lake Storage

The screenshot shows the Azure portal interface for creating a storage account. The account name is 'healthprojectadls'. Key details shown include:

- Resource group:** health-project-rg
- Location:** ukouth
- Subscription:** Shree-WP
- Performance:** Standard
- Replication:** Locally-redundant storage (LRS)
- Account kind:** StorageV2 (general purpose v2)
- Provisioning state:** Succeeded
- Created:** 5/29/2024, 10:50:06 PM

Creating Azure Data Factory

The screenshot shows the Azure portal interface for an Azure Data Factory deployment. The deployment status is "Your deployment is complete". Deployment details include:

- Deployment name: Microsoft.DataFactory-20240529224953
- Subscription: Shree-WP
- Resource group: health-project-rg
- Start time: 29/05/2024, 22:50:42
- Correlation ID: 611265ad-7318-4e7a-945e-973ebad845eb

Configuring GitHub in ADF

Configure a repository

No Git repository configured

Specify the settings that you want to use when connecting to your repository.

Select repository Use repository link

Repository name: project_1

Collaboration branch: main

Publish branch: workspace_publish

Root folder: /

Custom comment: Use custom comment

Import existing resources: Import existing resources to repository

Import resource into this branch:

Apply Back Cancel

Repo Connected

Repository settings have been successfully updated

S

Creating Linked Service of HTTP and ADLS Service in ADF

New linked service

Azure Data Lake Storage Gen2 [Learn more](#)

Name *

Description

Connect via integration runtime * [AutoResolveIntegrationRuntime](#)

Authentication type [Account key](#)

Account selection method [From Azure subscription](#) [Enter manually](#)

Azure subscription [Shri-WP \(0318f44-7292-4754-b814-23e209fc950\)](#)

Storage account * [healthprojectadls](#)

Test connection [To linked service](#) [To file path](#)

Annotations [New](#)

Parameters [New](#)

Advanced [New](#)

Create **Back** **Connection successful** **Test connection** **Cancel**

Edit linked service

HTTP [Learn more](#)

Name *

Description

Connect via integration runtime * [AutoResolveIntegrationRuntime](#)

Base URL * <https://fbstorage.googleapis.com/>

Information will be sent to the URL specified. Please ensure you trust the URL entered.

Server certificate validation [Enable](#) [Disable](#)

Authentication type [Anonymous](#)

Auth headers [New](#)

Annotations [New](#)

Parameters [New](#)

Advanced [New](#)

Save **Cancel** **Test connection**

Creating Dataset for HTTP input and ADLS output

Factory Resources

- Pipelines: pl_copy_http_data
- Change Data Capture (preview): 0
- Datasets: ds_health_source_csv, ds_health_sink_csv
- Data flows: 0
- Power Query: 0

ds_health_sink_csv

Connection

- Linked service: ls_health_http
- Base URL: https://firebasestorage.googleapis.com/
- Relative URL: /v0/b/music-player-98871.appspot.com/o/
- Compression type: Select...
- Column delimiter: Comma (,)
- Row delimiter: Default (\r\n, or \n\r)
- Encoding: Default(UTF-8)
- Quote character: Double quote (")
- Escape character: Backslash (\)
- First row as header:
- Null value:

Factory Resources

- Pipelines: pl_copy_http_data
- Change Data Capture (preview): 0
- Datasets: ds_health_source_csv, ds_health_sink_csv
- Data flows: 0
- Power Query: 0

ds_health_sink_csv

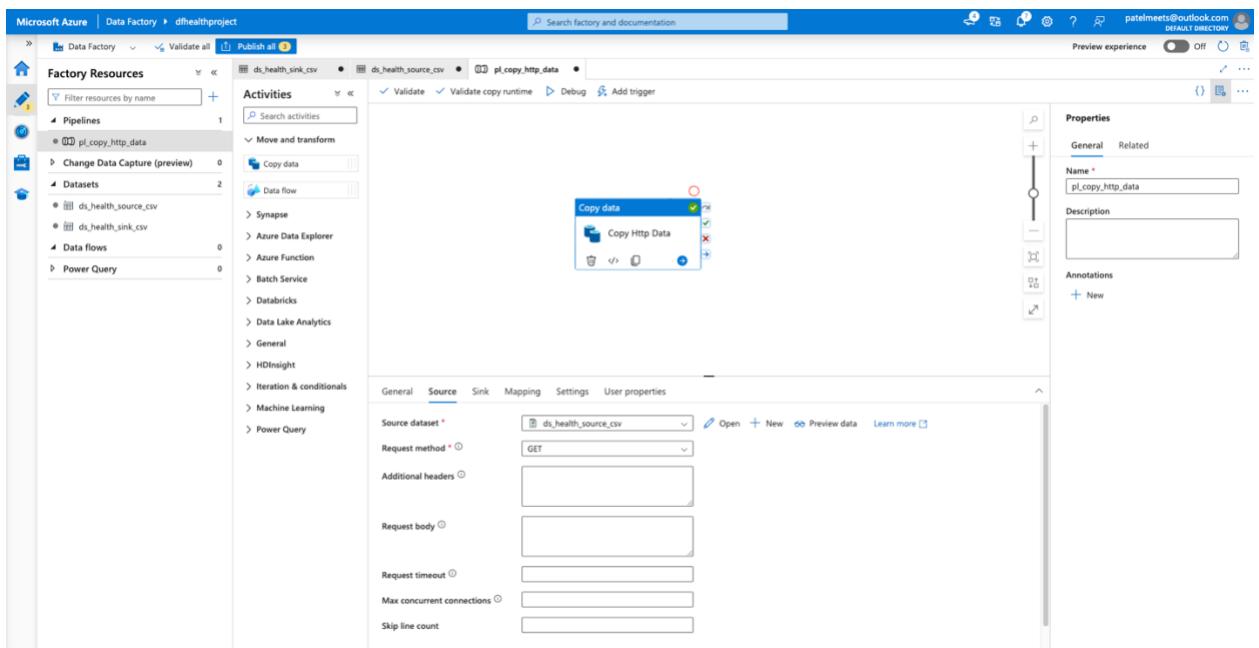
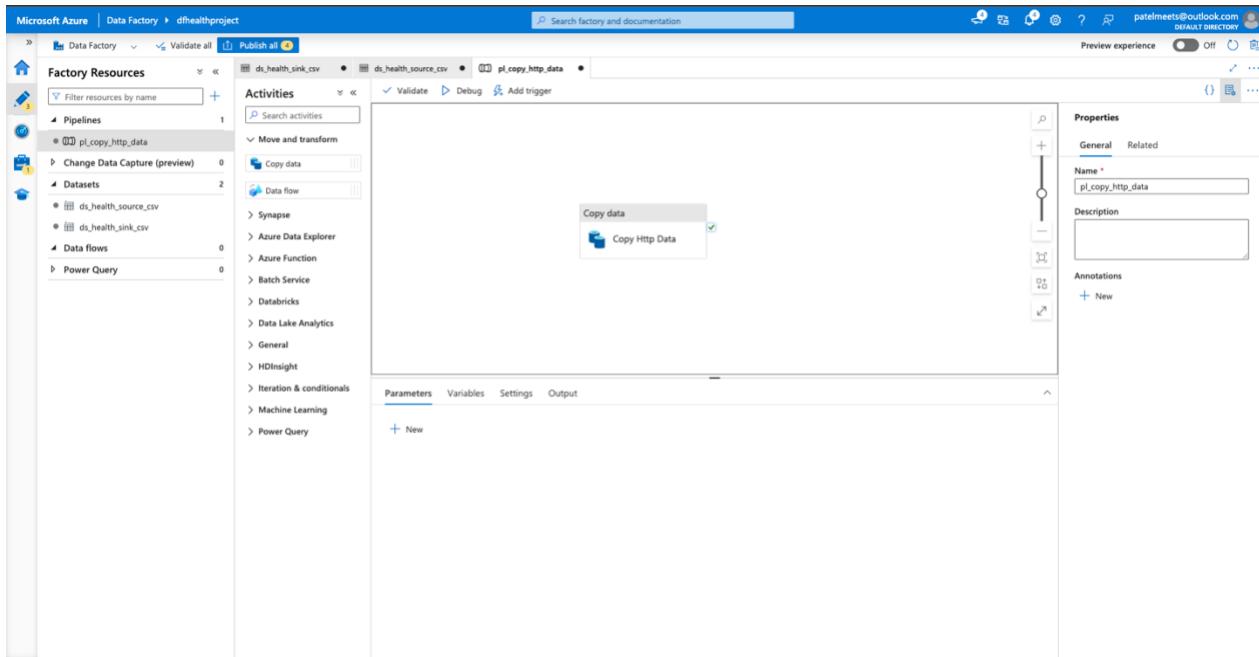
Properties

- Name: ds_health_sink_csv
- Description:
- Annotations: + New

Connection

- Linked service: ls_health_adls
- File path: healthcontainer / raw / cancer_data_raw.csv
- Compression type: Select...
- Column delimiter: Comma (,)
- Row delimiter: Default (\r\n, or \n\r)
- Encoding: Default(UTF-8)
- Quote character: Double quote (")
- Escape character: Backslash (\)
- First row as header:
- Null value:

Creating a Pipeline to transfer data from Firebase Cloud to ADLS using ADF



Pipeline Succeeded

The screenshot shows the Microsoft Azure Data Factory interface for the project 'dfhealthproject'. The left sidebar lists 'Factory Resources' including Pipelines, Datasets, Data flows, and Power Query. The main area displays the 'Activities' section with a 'Copy data' activity selected. The 'Properties' pane on the right shows the activity's name as 'pl_copy_http_data'. The 'Output' tab indicates a 'Pipeline run ID' of '5279a8ec-c74c-4b26-8d43-a88d03903e6d' and a 'Pipeline status' of 'Succeeded'. A table below shows the details of the single activity: 'Copy Http Data' with an 'Activity status' of 'Succeeded', 'Activity type' of 'Copy data', and a 'Run start' time of '5/30/2024, 12:15:44 AM'. The duration was '23s'.

Activity name	Activity status	Activity type	Run start	Duration	User properties
Copy Http Data	Succeeded	Copy data	5/30/2024, 12:15:44 AM	23s	AutoResolveIntegration

Transferred Data in ADLS

The screenshot shows the Microsoft Azure Storage Explorer interface. The left sidebar has a tree view with 'Overview' selected, and other options like 'Diagnose and solve problems' and 'Access Control (IAM)'. The main area is titled 'healthcontainer' and shows a single blob named 'cancer_data_raw'. The blob details are as follows:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
cancer_data_raw	30/05/2024, 00:17:32	Hot (Inferred)		Block blob	133.39 MB	Available

Databricks

Creating Databricks Workspace

retaildbrwp Azure Databricks Service

Overview

Essentials

- Status : Active
- Resource group : [retail-rg](#)
- Location : East US
- Subscription : [Shreehari-WP](#)
- Subscription ID : 9b093c33-a116-4e12-aa43-073a1c622e60
- Tags (edit) : [Add tags](#)

Managed Resource Group : [databricks-rg-retaildbrwp-nqf3ht5ign6rw](#)
URL : <https://adb-3107986375698214.14.azuredatabricks.net>
Pricing Tier : [Standard \(Apache Spark, Secure with Microsoft Entra ID\)](#) (Click to...)

Launch Workspace

Upgrade to Premium

Documentation

Getting Started

Import Data from File

Import Data from Azure Storage

Notebook

Admin Guide

Link Azure ML workspace

Creating Key vault

The screenshot shows the Microsoft Azure Key vaults interface. The top navigation bar includes 'Microsoft Azure', 'Upgrade', a search bar, and user information ('Shrijit74@outlook.com DEFAULT DIRECTORY (SHRIJIT74...)'). The main page title is 'Key vaults' under 'Default Directory (Shrijit74outlook.onmicrosoft.com)'. A sidebar on the left lists vaults: 'health-key-vault' (selected), 'retail-key-vault', and others. The central 'Overview' section for 'health-key-vault' displays details like Resource group (health-rg), Location (East US), Subscription (Shrijit-Wp), and Tags. It also shows 'Essentials' such as Vault URI, Sku (Standard), and Soft-delete status. Below this are sections for 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Events', 'Objects', 'Settings', 'Monitoring', 'Automation', and 'Help'. At the bottom, there are links for 'Get started', 'Properties', 'Monitoring', 'Tools + SDKs', and 'Tutorials'. A call-to-action 'Manage keys and secrets used by apps and services' is present, along with a note about best practices for vault usage. Three quick actions are shown: 'Control access to key vault' (Assign access policy and), 'Enable logging and set up alerts' (Enable logging to monitor how), and 'Turn on recovery options' (For protection against accidental).

Creating App in Microsoft Intra ID

Overview

Display name : [health-project](#)

Application (client) ID : e6a15e72-4a1b-46c4-91ca-06faf86a7da

Object ID : b0fe4617-3dc1-4fcf-a360-b412b5333392

Directory (tenant) ID : 01d0a7d7-12e7-422a-9a39-9c406c7e4496

Client credentials : [0_certificate_1 secret](#)

Redirect URIs : [Add a Redirect URI](#)

Application ID URI : [Add an Application ID URI](#)

Supported account types : [My organization only](#)

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)

[Get Started](#) [Documentation](#)

Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. [Learn more](#)

Creating secrets in Key Vault

The screenshot shows the Microsoft Azure Key Vault interface for the 'health-key-vault' resource. The left sidebar has a 'Secrets' section selected. The main area displays a table of secrets:

Name	Type	Status	Expiration date
secret-id		✓ Enabled	
tenant-id		✓ Enabled	
client-id		✓ Enabled	

A message at the top right states: 'The secret 'secret-id' has been successfully created.'

Giving RBAC Permission of Storage Blob Contributor to ADLS

The screenshot shows the Microsoft Azure portal interface for managing role assignments. On the left, the 'Add role assignment' dialog is open, showing the 'Members' tab selected. It lists a 'Selected role' of 'Storage Blob Data Contributor' and 'Assign access to' as 'User, group, or service principal'. Below these are sections for 'Members' and 'Description'. At the bottom are 'Review + assign', 'Previous', and 'Next' buttons. On the right, a 'Select members' modal is displayed, showing a search bar with 'heal' typed in, a message 'No users, groups, or service principals found.', and a 'Selected members:' section containing 'health-project' with a 'Remove' link. At the bottom of the modal are 'Select' and 'Close' buttons.

Creating Secret Scope

Scope Name

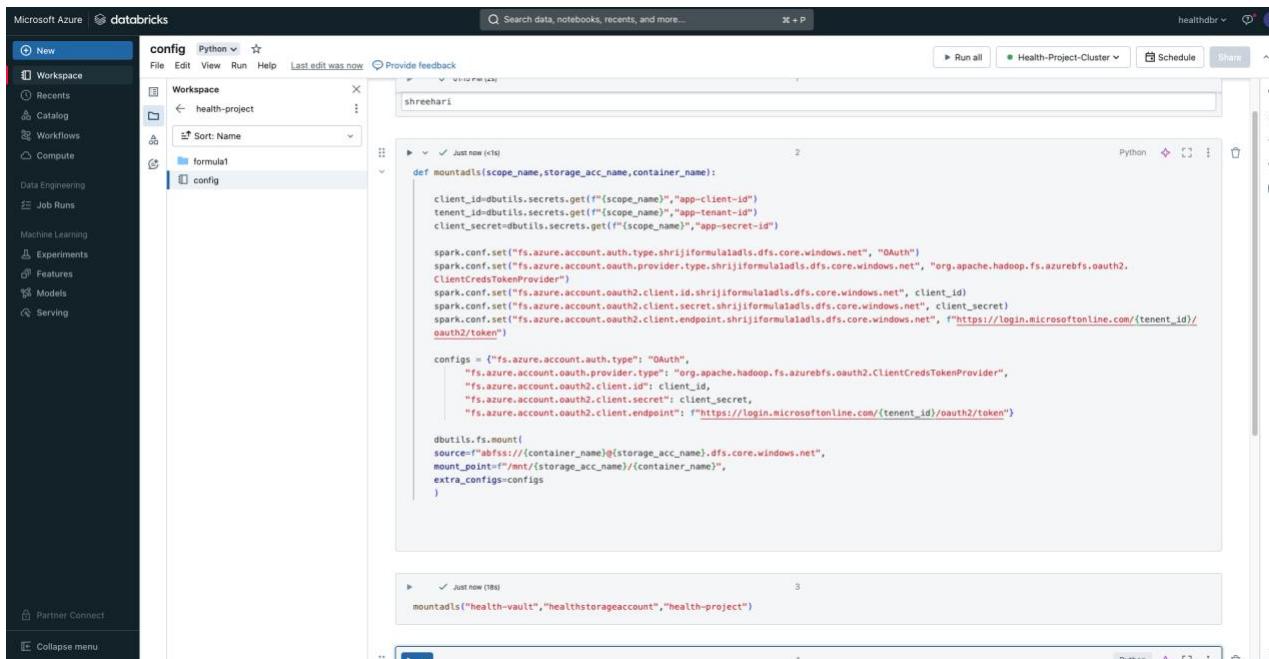
Manage Principal

Azure Key Vault

DNS Name

Resource ID

Mounting ADLS to Databricks Workspace



The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. The main area displays a notebook titled 'config' in Python. The code in the notebook is as follows:

```
def mountadls(scope_name,storage_acc_name,container_name):
    client_id=dbutils.secrets.get(scope_name,"app-client-id")
    tenant_id=dbutils.secrets.get(scope_name,"app-tenant-id")
    client_secret=dbutils.secrets.get(scope_name,"app-secret-id")

    spark.conf.set("fs.azure.account.auth.type", "OAuth")
    spark.conf.set("fs.azure.account.oauth.provider.type", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
    spark.conf.set("fs.azure.account.oauth2.client.id", client_id)
    spark.conf.set("fs.azure.account.oauth2.client.secret", client_secret)
    spark.conf.set("fs.azure.account.oauth2.client.endpoint", "https://login.microsoftonline.com/{tenant_id}/oauth2/token")

    configs = ("fs.azure.account.auth.type": "OAuth",
               "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
               "fs.azure.account.oauth2.client.id": client_id,
               "fs.azure.account.oauth2.client.secret": client_secret,
               "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/{tenant_id}/oauth2/token")

    dbutils.fs.mount(
        source=f"abfss://{container_name}@{storage_acc_name}.dfs.core.windows.net",
        mount_point=f"/mnt/{storage_acc_name}/{container_name}",
        extra_configs=configs
    )

mountedls("health-vault","healthstorageaccount","health-project")
```

Creating a config file

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. The main area displays a notebook titled 'config' in Python. The notebook contains two cells:

- Cell 1:

```
base_url = "dbfs:/mnt/healthstorageaccount/health-project"
```
- Cell 2:

```
display(dbutils.fs.mounts())
```

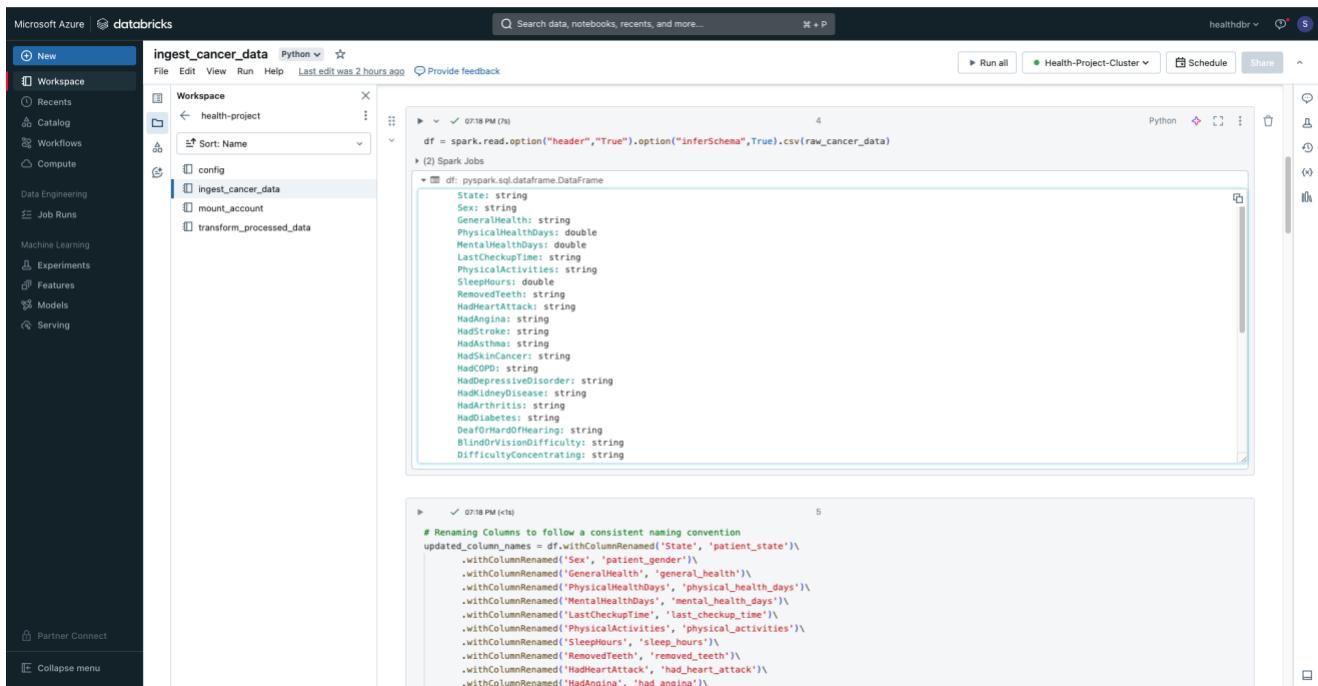
Cell 2 has a table output showing the following data:

mountPoint	source	encryptionType
/databricks-datasets	databricks-datasets	
/Volumes	UnityCatalogVolumes	
/mnt/healthstorageaccount/health-proje...	abfss://health-project@healthstorageaccount.dfs.core.windows.net	
/databricks/mflow-tracking	databricks/mflow-tracking	
/databricks-results	databricks-results	
/databricks/mflow-registry	databricks/mflow-registry	
/Volume	DbfsReserved	
/volumes	DbfsReserved	
/	DatabricksRoot	
/volume	DbfsReserved	

At the bottom of the notebook interface, there are instructions: "[Shift+Enter] to run and move to next cell" and "[Esc H] to see all keyboard shortcuts".

Data Cleaning

Reading Data through PySpark



The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. The main area displays a notebook titled "ingest_cancer_data" in Python.

Code Cell 1:

```
df = spark.read.option("header","True").option("inferSchema",True).csv(raw_cancer_data)
```

This cell reads a CSV file named "raw_cancer_data" into a DataFrame named "df". The code uses the `spark.read` API with options for header detection and schema inference.

Code Cell 2:

```
# Renaming Columns to follow a consistent naming convention
updated_column_names = df.withColumnRenamed('State', 'patient_state')\
    .withColumnRenamed('Sex', 'patient_gender')\
    .withColumnRenamed('GeneralHealth', 'general_health')\
    .withColumnRenamed('PhysicalHealthDays', 'physical_health_days')\
    .withColumnRenamed('MentalHealthDays', 'mental_health_days')\
    .withColumnRenamed('LastCheckupTime', 'last_checkup_time')\
    .withColumnRenamed('PhysicalActivities', 'physical_activities')\
    .withColumnRenamed('Sleephours', 'sleep_hours')\
    .withColumnRenamed('RemovedTeeth', 'removed_teeth')\
    .withColumnRenamed('HadheartAttack', 'had_heart_attack')\
    .withColumnRenamed('HadAngina', 'had_angina')
```

This cell renames columns in the DataFrame "df" to follow a consistent naming convention. It uses the `withColumnRenamed` method to change column names like "State", "Sex", and "GeneralHealth" to "patient_state", "patient_gender", and "general_health" respectively.

Handling Missing Values

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. The main area displays a Python notebook titled 'ingest_cancer_data'. The notebook's code cell contains the following Python code:

```

#Removing Missing Values

mode_gender = updated_column_names.groupBy('patient_gender').count().orderBy(desc("count")).first()[0]
mode_state = updated_column_names.groupBy('patient_state').count().orderBy(desc("count")).first()[0]
mode_general_health = updated_column_names.groupBy('general_health').count().orderBy(desc("count")).first()[0]
mode_race_ethnicity_category = updated_column_names.groupBy('race_ethnicity_category').count().orderBy(desc("count")).first()[0]
mode_race_age_category = updated_column_names.groupBy('age_category').count().orderBy(desc("count")).first()[0]

mean_height = updated_column_names.select(round(mean(col('height_in_meters')))).first()[0]
mean_weight = updated_column_names.select(round(mean(col('weight_in_kilograms')))).first()[0]
mean_bmi = updated_column_names.select(round(mean(col('bmi')))).first()[0]

removed_missing_data = updated_column_names.fillna({patient_state:mode_state,'patient_gender':mode_gender,'general_health':mode_general_health,
'physical_health_days':10,'mental_health_days':10,'last_checkup_time':'Unknown','physical_activities':'No','sleep_hours':7,'had_heart_attack':'No',
'had_angina':'No','had_stroke':'No','had_asthma':'No','had_diabetes':'No','had_skin_cancer':'No','had_copd':'No','had_depressive_disorder':'No',
'had_kidney_disease':'No','had_arthritis':'No','blind_or_vision_difficulty':'No','difficulty_concentrating':'No','difficulty_walking':'No',
'difficulty_dressing_bathing':'No','difficulty_errands':'No','chest_scan':'No','alcohol_drinkers':'No','hiv_testing':'No','flu_vax_last_12':'No',
'pneumo_vax_ever':'No','high_risk_last_year':'No','covid_pos':'No','smoker_status':'Never smoked','e-cigarette_usage':'Never used e-cigarettes in my
entire life','race_ethnicity_category':mode_race_ethnicity_category,'age_category':mode_race_age_category,'height_in_meters':mean_height,
'weight_in_kilograms':mean_weight,'tetanus_last_10_tdap':'Unknown','deaf_or_hard_of_hearing':'No','bmi':mean_bmi})

```

The notebook also contains a command to display the count of heart attacks by age category:

```
display(removed_missing_data.groupBy("age_category").agg(count.when(col('had_heart_attack')=='Yes',1)).alias("heart_attack_count"))
```

A table below the code cell shows the resulting data:

age_category	heart_attack_count
Age 45 to 49	728
Age 25 to 29	119

Dropping Unwanted Columns using .drop func

```

File Edit View Run Help Last edit was now Provide feedback
Run all Health-Project-Cluster Schedule Share
ingest_cancer_data Python 9
removed_column_data = removed_missing_data.drop(col('removed_teeth'))
removed_column_data: pyspark.sql.dataframe.DataFrame = [patient_state: string, patient_gender: string ... 37 more fields]

10
display(removed_column_data)
# (1) Spark.Jobs
Table + New result table: ON □ □ □
patient_state patient_gender general_health physical_health_days mental_health_days last_checkup_time
1 Alabama Female Very good 0 0 Within past year (anytime less than 12 months ago)
2 Alabama Female Excellent 0 0 Unknown
3 Alabama Female Very good 2 3 Within past year (anytime less than 12 months ago)
4 Alabama Female Excellent 0 0 Within past year (anytime less than 12 months ago)
5 Alabama Female Fair 2 0 Within past year (anytime less than 12 months ago)
6 Alabama Male Poor 1 0 Within past year (anytime less than 12 months ago)
7 Alabama Female Very good 0 0 Within past year (anytime less than 12 months ago)
8 Alabama Female Good 0 0 Within past year (anytime less than 12 months ago)
9 Alabama Female Good 0 0 Within past year (anytime less than 12 months ago)
10 Alabama Female Good 1 0 Within past year (anytime less than 12 months ago)
11 Alabama Female Fair 8 9 Within past year (anytime less than 12 months ago)
12 Alabama Female Good 0 0 Within past year (anytime less than 12 months ago)
13 Alabama Male Fair 5 0 Within past year (anytime less than 12 months ago)
14 Alabama Male Very good 0 0 Within past year (anytime less than 12 months ago)

5,438+ rows | Truncated data due to byte limit | 1.99 seconds runtime
Refreshed 2 hours ago
11
# Check for logical consistency in AgeCategory
valid_age_categories = ["Age 80 or older", "Age 75 to 79", "Age 70 to 74", "Age 65 to 69",

```

Making Data Consistent

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. The main area displays three notebooks:

- ingest_cancer_data**: A Python notebook containing code to check logical consistency in AgeCategory and remove rows where bmi is less than 10 or greater than 60. It also adds a timestamp column to the final data.
- final_data**: A Python notebook showing the final data frame with columns patient_state, patient_gender, and timestamp.
- heart_attack_count**: A Python notebook displaying a table of age categories and their corresponding heart attack counts.

The table from the heart_attack_count notebook is as follows:

age_category	heart_attack_count
Age 45 to 49	728
Age 25 to 29	119
Age 70 to 74	4216
Age 55 to 59	1960
Age 60 to 64	2890
Age 50 to 54	1250
Age 35 to 39	318
Age 30 to 34	187
Age 80 or older	5024
Age 40 to 44	445
Age 75 to 79	3789
Age 65 to 69	4182

Writing Data to ADLS in Parquet Format

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like 'New', 'Workspace', 'Recents', 'Catalog', 'Workflows', 'Compute', 'Data Engineering', 'Job Runs', 'Machine Learning', 'Experiments', 'Features', 'Models', and 'Serving'. The main area shows a notebook titled 'ingest_cancer_data' in Python. The notebook contains the following content:

```
File Edit View Run Help Last edit was now Provide feedback
Search data, notebooks, recents, and more...
Run all Health-Project-Cluster Schedule Share
ingest_cancer_data Python
Workspace
health-project
Sort: Name
config
ingest_cancer_data
mount_account
transform_processed_data
(2) Spark Jobs
Table + New result table: ON
age_category heart_attack_count
1 Age 45 to 49 728
2 Age 25 to 29 119
3 Age 70 to 74 4216
4 Age 55 to 59 1960
5 Age 60 to 64 2890
6 Age 50 to 54 1250
7 Age 35 to 39 318
8 Age 30 to 34 187
9 Age 80 or older 5024
10 Age 40 to 44 445
11 Age 75 to 79 3789
12 Age 65 to 69 4182
12 rows | 2.36 seconds runtime
Refreshed 2 hours ago
14
final_data.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/processed/cancer')
(1) Spark Jobs
15
[Shift+Enter] to run and move to next cell
[Esc M] to see all keyboard shortcuts
```

Parquet Processed Data in ADLS

The screenshot shows the Microsoft Azure Storage Explorer interface. The top navigation bar includes 'Microsoft Azure', 'Upgrade', a search bar ('Search resources, services, and docs (G+ /)'), and user information ('Shriji74@outlook.com DEFAULT DIRECTORY (SHRIJI740...)'). Below the navigation is a breadcrumb trail: 'Home > Storage accounts > healthstorageaccount | Containers >'. The main area displays the 'health-project' container, which is a 'Container'. The left sidebar shows 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. The main content area shows a table of blobs:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state	
[...]						***	
_committed_1048419511357261068	01/06/2024, 17:05:08	Hot (Inferred)		Block blob	815 B	Available	***
_committed_2580617460693770546	01/06/2024, 18:06:54	Hot (Inferred)		Block blob	823 B	Available	***
_committed_3317609724609499390	01/06/2024, 17:12:43	Hot (Inferred)		Block blob	819 B	Available	***
_committed_973792954262681395	31/05/2024, 21:27:59	Hot (Inferred)		Block blob	830 B	Available	***
_started_2580617460693770546	01/06/2024, 18:06:49	Hot (Inferred)		Block blob	0 B	Available	***
part-00000-tid-2580617460693770546...	01/06/2024, 18:06:51	Hot (Inferred)		Block blob	1.21 MiB	Available	***
part-00001-tid-2580617460693770546...	01/06/2024, 18:06:52	Hot (Inferred)		Block blob	1.18 MiB	Available	***
part-00002-tid-2580617460693770546...	01/06/2024, 18:06:51	Hot (Inferred)		Block blob	1.19 MiB	Available	***
part-00003-tid-2580617460693770546...	01/06/2024, 18:06:53	Hot (Inferred)		Block blob	1.06 MiB	Available	***

Data Transformation

Reading Parquet Data using PySpark

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes 'New', 'Workspace', 'Recents', 'Catalog', 'Workflows', 'Compute', 'Data Engineering', 'Job Runs', 'Machine Learning', 'Experiments', 'Features', 'Models', and 'Serving'. A 'Partner Connect' button and a 'Collapse menu' button are also present. The main area displays a notebook titled 'transform_processed_data' in Python. The notebook contains four runs:

- Run 1: '%run "/config"' - A table titled 'Table' is shown with the following data:

#	mountPoint	source	encryptionType
1	/databricks-datasets	databricks-datasets	
2	/Volumes	UnityCatalogVolumes	
3	/mnt/healthstorageaccount/health-project/	abfss://health-project@healthstorageaccount.dfs.core.windows.net	
4	/databricks/mflow-tracking	databricks/mflow-tracking	
5	/databricks-results	databricks-results	
6	/databricks/mflow-registry	databricks/mflow-registry	
7	/Volume	DbsReserved	
8	/volumes	DbsReserved	
9	/	DatabricksRoot	
10	/volume	DbsReserved	

10 rows | 0.59 seconds runtime

- Run 2: PySpark code snippet:

```
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

- Run 3: PySpark code snippet:

```
df = spark.read.option("header",True).parquet('/mnt/healthstorageaccount/health-project/processed/cancer/')
```

(1) Spark Jobs

```
df: pyspark.sql.dataframe.DataFrame = [patient_state: string, patient_gender: string ... 38 more fields]
```

- Run 4: PySpark code snippet:

Aggregating Data of BMI and Sleep Hours

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes sections for New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. A 'Partner Connect' button and a 'Collapse menu' link are also present.

The main area displays two notebooks:

- Notebook 1 (Top):** Titled 'transform_processed_data'. It contains the following code:

```
df_agg_bmi_sleep = df.groupby('patient_state').agg(round(avg('bmi'),2).alias("avg_bmi"),sum("sleep_hours").alias("total_sleep_hours"))
display(df_agg_bmi_sleep)
```

A table titled 'Table' is shown with the following data:

#	patient_state	1.2 avg_bmi	1.2 total_sleep_hours
1	Hawaii	27	52676
2	Arkansas	29	37391
3	Connecticut	28	68025
4	Illinois	29	28141
5	District of Columbia	27	22809
6	Delaware	29	27940
7	Alaska	28	41330
8	Georgia	29	64812
9	Alabama	29	31578
10	Arizona	28	72263
11	Iowa	29	63222
12	Florida	28	94500
13	Indiana	29	72884
14	Idaho	28	44387
15	California	28	76528

Runtime information: 54 rows | 0.28 seconds runtime. Last updated: Refreshed 2 hours ago.
- Notebook 2 (Bottom):** Titled 'transform_processed_data'. It contains the following code:

```
df_agg_heartattack_agegroup = df.groupby('age_category').agg(count(when(col('had_heart_attack')=='Yes',1)).alias("heart_attack_count")).orderBy('age_category')
display(df_agg_heartattack_agegroup)
```

A table titled 'Table' is shown with the following data:

#	age_category	heart_attack_count
1	0-14	1
2	15-24	1
3	25-34	1
4	35-44	1
5	45-54	1
6	55-64	1
7	65-74	1
8	75-84	1
9	85+	1

Aggregating Data of Heart Attack and Age group

The screenshot shows the Databricks workspace interface. On the left is the sidebar with options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. The main area has two notebooks running.

Notebook 1: transform_processed_data

```

07:31 PM (c1)
df_agg_heartattack_agegroup = df.groupBy("age_category").agg(count(when(col('had_heart_attack')=='Yes',1)).alias("heart_attack_count")).orderBy('age_category')
display(df_agg_heartattack_agegroup)
(2) Spark Jobs
df_agg_heartattack_agegroup: pyspark.sql.dataframe.DataFrame = [age_category: string, heart_attack_count: long]

```

Table:

age_category	heart_attack_count
Age 25 to 29	119
Age 30 to 34	187
Age 35 to 39	318
Age 40 to 44	445
Age 45 to 49	728
Age 50 to 54	1250
Age 55 to 59	1960
Age 60 to 64	2890
Age 65 to 69	4182
Age 70 to 74	4216
Age 75 to 79	3789
Age 80 or older	5024

12 rows | 0.30 seconds runtime
Refreshed 2 hours ago

Notebook 2: (2) Spark Jobs

```

07:31 PM (c1)
df_agg_physhealth_gender = df.groupBy('patient_gender').agg(sum('physical_health_days').alias("total_physical_health_days"))
display(df_agg_physhealth_gender)
(2) Spark Jobs
df_agg_physhealth_gender: pyspark.sql.dataframe.DataFrame = [patient_gender: string, total_physical_health_days: double]

```

Table:

patient_gender	total_physical_health_days
Female	104092

New result table: ON

Aggregating Data of Gender physical days

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Features, Models, and Serving. The main area shows two notebooks running:

- transform_processed_data (Python):** This notebook runs a command to aggregate physical health days by gender. The resulting DataFrame is displayed below:

patient_gender	total_physical_health_days
Female	1080882
Male	807006

Runtime: 0.20 seconds, Refreshed 2 hours ago.
- transform_processed_data (Python):** This notebook runs a command to aggregate mental health days by gender. The resulting DataFrame is displayed below:

patient_gender	total_mental_health_days
Female	1158138
Male	752982

Runtime: 0.20 seconds, Refreshed 2 hours ago.

Writing this Aggregated Data in different Directory

The screenshot shows the Microsoft Azure Databricks workspace. A notebook titled "transform_processed_data" is open in Python mode. The notebook contains the following code:

```
df_agg_bmi_sleep.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_bmi_sleep')
df_agg_physhealth_gender.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_physhealth_gender')
df_agg_mentalhealth_gender.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_mentalhealth_gender')
df_agg_heartattack_agegroup.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_heartattack_agegroup')

# [10] Spark Jobs
```

Below the code, there is a table with two rows of data:

	Female	Male	1158138
1	Female	Male	1158138
2	Male	Male	752982

Information at the bottom of the notebook includes:

- [Shift+Enter] to run and move to next cell
- [Esc H] to see all keyboard shortcuts

Partitioning the Data and Writing in ADLS in Parquet Format

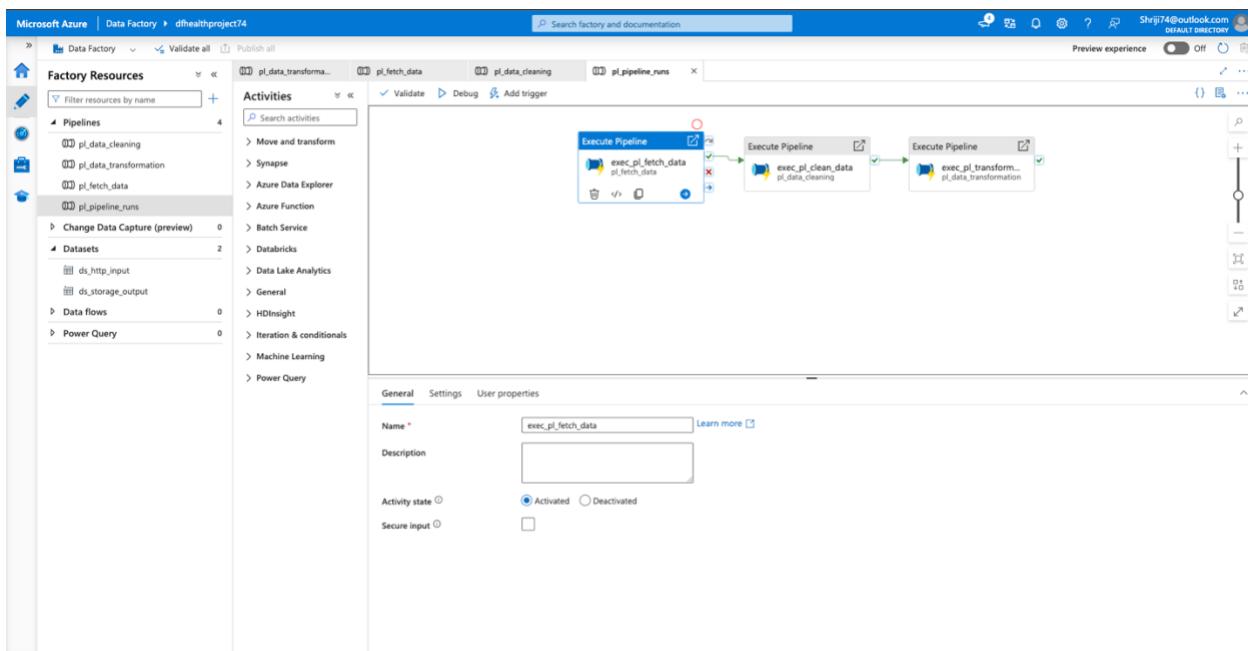
The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Experiments, Features, Models, and Serving. The main area displays a notebook titled "transform_processed_data" in Python. The notebook contains two code cells. The first cell, labeled "9", contains the following code:df_agg_bmi_sleep.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_bmi_sleep')
df_agg_physhealth_gender.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_physhealth_gender')
df_agg_mentalhealth_gender.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_mentalhealth_gender')
df_agg_hearattack_agegroup.write.mode('overwrite').parquet('/mnt/healthstorageaccount/health-project/transformed/Agg_hearattack_agegroup')The second cell, labeled "10", contains:df.write.mode('overwrite').partitionBy('patient_state').parquet('/mnt/healthstorageaccount/health-project/transformed/Patient_State_Data')
df.write.mode('overwrite').partitionBy('patient_gender').parquet('/mnt/healthstorageaccount/health-project/transformed/Patient_Gender_Data')
df.write.mode('overwrite').partitionBy('age_category').parquet('/mnt/healthstorageaccount/health-project/transformed/Age_Category_Data')
df.write.mode('overwrite').partitionBy('had_heart_attack').parquet('/mnt/healthstorageaccount/health-project/transformed/HeartAttack_Data')Both cells show a runtime of 0.20 seconds. A data preview table shows 2 rows with columns Female, Male, and IDs 1158138 and 752982. The notebook was last edited 2 hours ago and was refreshed 2 hours ago. The status bar at the bottom indicates "[Shift+Enter] to run and move to next cell" and "[Esc H] to see all keyboard shortcuts".

Creating Databricks Notebook Pipeline of Data cleaning and Data Transformation

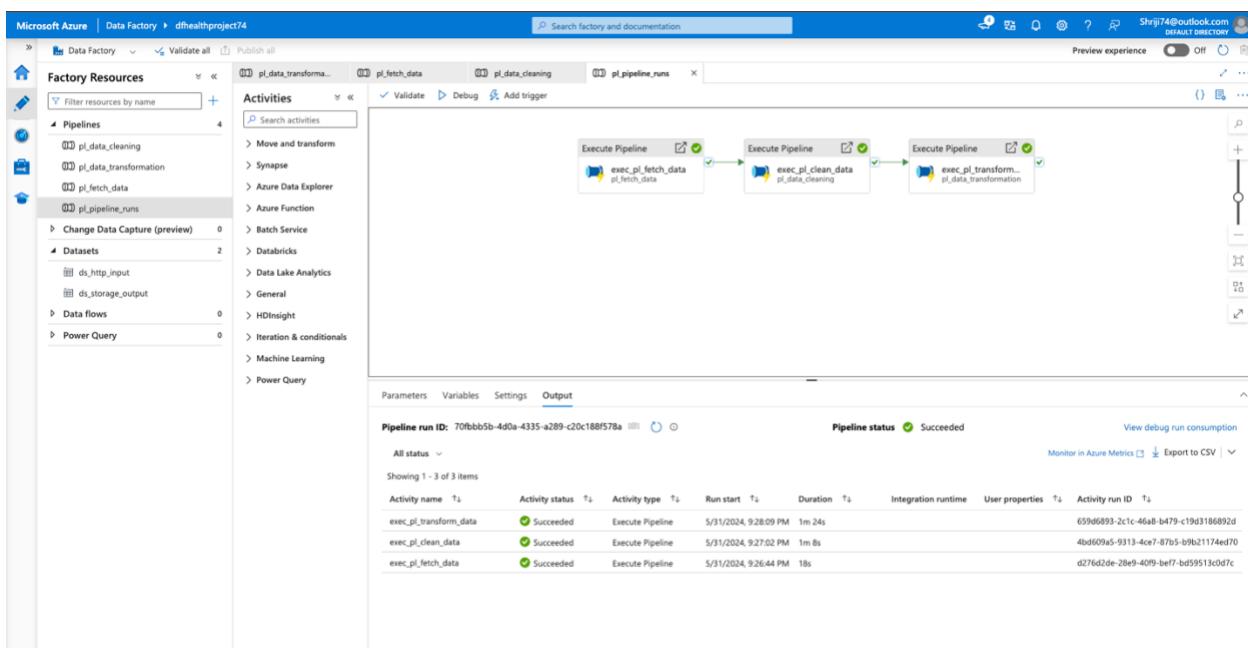
The screenshot shows the Microsoft Azure Data Factory pipeline editor interface. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, Data flows, and Power Query. In the center, the 'Activities' pane shows three parallel activities: 'pl_data_transform...', 'pl_fetch_data', and 'pl_data_cleaning'. The 'pl_data_cleaning' activity is currently selected. A preview window displays a 'Notebook' icon with the name 'data_cleaning_notebook'. Below the preview, the 'General' tab of the activity configuration pane is visible, showing settings like Name (data_cleaning_notebook), Activity state (Activated), and Timeout (0.12:00:00). Other tabs include 'Azure Databricks', 'Settings', and 'User properties'.

This screenshot shows the same Azure Data Factory interface, but the 'pl_data_cleaning' activity has been replaced by a new activity named 'pl_data_transformation'. The preview window now displays a 'Notebook' icon with the name 'data_transformation_notebook'. The 'General' tab of this activity is shown, with the 'Databricks linked service' dropdown set to 'AzureDatabricks1'. Other tabs like 'Azure Databricks', 'Settings', and 'User properties' are also present.

Creating a Pipeline to Execute all three pipelines.



Pipeline Succeeded



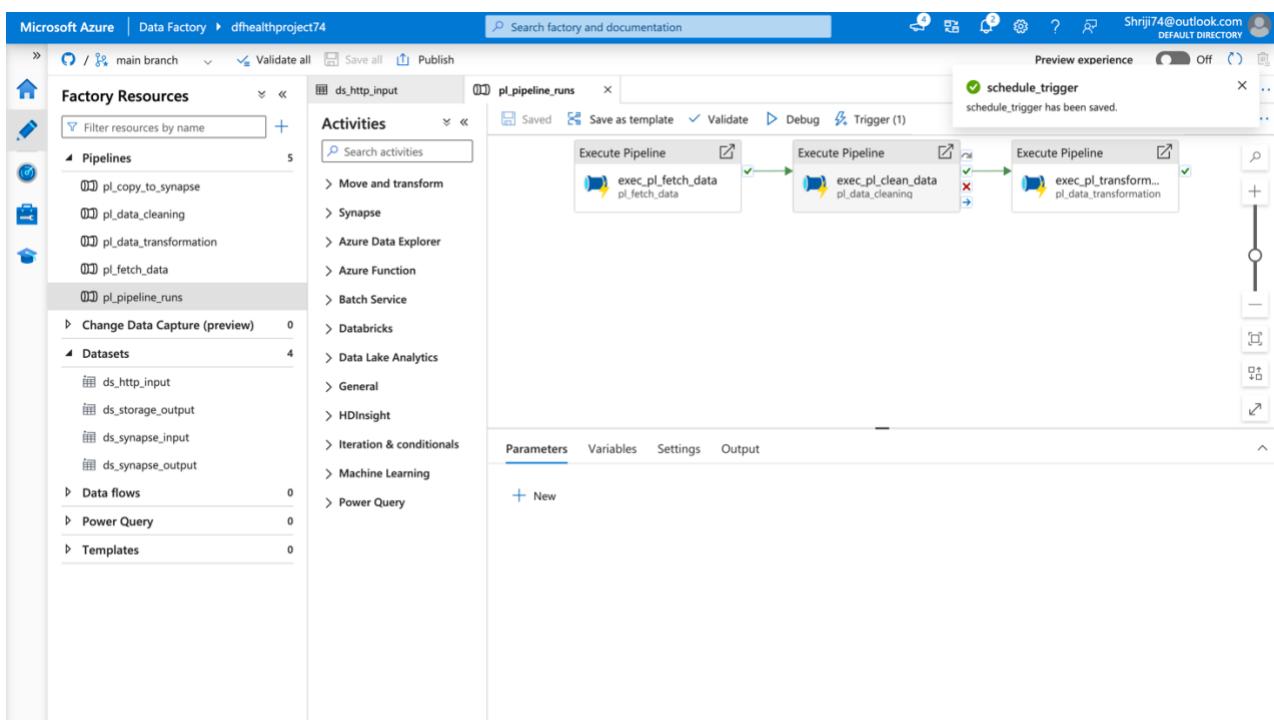
Creating Schedule Trigger

The screenshot shows the Microsoft Azure Data Factory interface for creating a new trigger. The left sidebar navigation includes General, Factory settings, Connections (Linked services, Integration runtimes, Microsoft Purview), Source control (Git configuration, ARM template), Author (Triggers, Global parameters, Data flow libraries), Security (Credentials, Customer managed key, Outbound rules, Managed private endpoints), Workflow orchestration manager, and Apache Airflow. The 'Triggers' option under Author is selected.

The main content area displays the 'New trigger' dialog. The 'Name' field is set to 'schedule_trigger'. The 'Type' dropdown is set to 'Schedule'. The 'Start date' is set to '6/7/2024, 5:56:10 AM'. The 'Time zone' is set to 'Eastern Time (US & Canada) (UTC-4)'. A note indicates that this time zone observes daylight savings, so the trigger will auto-adjust for one hour difference. The 'Recurrence' section shows 'Every 15 Minute(s)'. There is an unchecked checkbox for 'Specify an end date'. Under 'Annotations', there is a '+ New' button and a checked checkbox for 'Start trigger on creation'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

Adding Schedule Trigger to Pipeline

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar shows a list of pipelines, datasets, data flows, Power Query, and Templates. The 'Pipelines' section is expanded, showing five pipelines: pl_copy_to_synapse, pl_data_cleaning, pl_data_transformation, pl_fetch_data, and pl_pipeline_runs. The 'Activities' pane on the right lists various Azure services and connectors. A modal dialog titled 'Add triggers' is open, prompting the user to choose a trigger type. The pipeline itself is visible in the background, showing a sequence of activities: Execute Pipeline (exec_pl_fetch_data), Move and transform, Synapse, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, HDInsight, Iteration & conditionals, Machine Learning, and Power Query.



Azure Synapse Analytics

Creating Synapse Workspace

The screenshot shows the Microsoft Azure portal interface for a Synapse workspace named 'health-synapse'. The top navigation bar includes 'Microsoft Azure', 'Upgrade', a search bar, and user information for 'Shrijit74@outlook.com'.

Overview section:

- Essentials** table:

Resource group (move)	: health-rg	Networking	: Show firewall settings
Status	: Succeeded	Primary ADLS Gen2 acco...	: https://healthstorageaccount.dfs.core.windows.net
Location	: East US	Primary ADLS Gen2 file s...	: health-project
Subscription (move)	: Shrijit74	SQL admin username	: shrijit
Subscription ID	: 9b093c33-a116-4e12-aa43-073a1c62ze60	SQL Microsoft Entra admin	: live.com#Shrijit74@outlook.com
Managed virtual network	: No	Dedicated SQL endpoint	: health-synapse.sql.azuresynapse.net
Managed identity object	: 235f0ea2-6988-4389-8ab9-c0b49be07d9c	Serverless SQL endpoint	: health-synapse-on-demand.sql.azuresynapse.net
Workspace web URL	: https://web.azure-synapse.net/workspace=%25b093c33-a116-4e12-aa43-073a1c62	Development endpoint	: https://health-synapse.dev.azuresynapse.net
- Getting started** section:
 - Open Synapse Studio**: Start building your fully-integrated analytics solution and unlock new insights. [Open](#)
 - Read documentation**: Learn how to be productive quickly. Explore concepts, tutorials, and samples. [Learn more](#)

Analytics pools section:

Name	Type	Size
Built-in	Serverless	Auto
Apache Spark pools		
Data Explorer pools		

Creating Table in Dedicated SQL Pool

The screenshot shows the Microsoft Azure Synapse Analytics interface. The top navigation bar includes 'Microsoft Azure', 'Synapse Analytics', 'health-synapse', and a user account 'Shrij74@outlook.com'. The main workspace is titled 'Develop' and shows a file structure under 'SQL scripts'. A 'SQL script 2' file is currently selected. The code editor displays the following T-SQL script:

```

1 -- Create the internal table
2 CREATE TABLE dbo.heart_disease_data_internal (
3     patient_state NVARCHAR(100),
4     patient_gender NVARCHAR(100),
5     general_health NVARCHAR(100),
6     physical_health_days INT,
7     mental_health_days INT,
8     usd_consumption NVARCHAR(100),
9     physical_activities NVARCHAR(100),
10    sleep_hours FLOAT,
11    removed_teeth NVARCHAR(100),
12    had_heart_attack NVARCHAR(100),
13    had_angina NVARCHAR(100),
14    had_stroke NVARCHAR(100),
15    had_diabetes NVARCHAR(100),
16    had_skin_cancer NVARCHAR(100),
17    had_copd NVARCHAR(100),
18    had_depressive_disorder NVARCHAR(100),
19    had_kidney_disease NVARCHAR(100),
20    had_arthritis NVARCHAR(100),
21    had_diabetes NVARCHAR(100),
22    sex_genotype_of_patient NVARCHAR(100),
23    blind_or_vision_difficulty NVARCHAR(100),
24    difficulty_concentrating NVARCHAR(100),
25    difficulty_walking NVARCHAR(100),
26    difficulty_dressing_bathing NVARCHAR(100),
27    difficulty_errands NVARCHAR(100),
28    smoker_status NVARCHAR(100),
29    ecigarette_usage NVARCHAR(100),
30    chest_pain_type NVARCHAR(100),
31    race_ethnicity_category NVARCHAR(100),
32    age_category NVARCHAR(100),
33    height_in_meters FLOAT,
34    weight_in_kilograms FLOAT,
35    bmi FLOAT,
36    alcohol_drinkers NVARCHAR(100),
37    hiv_testing NVARCHAR(100),

```

The right-hand panel displays the 'Properties' for 'SQL script 2', showing the type as 'sql script' and results per query set to 'First 5000 rows (default)'. At the bottom of the interface, a message indicates '00:00:02 Query executed successfully.'

Creating Linked Service in Synapse

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the navigation menu includes General, Connections, and Linked services. Under General, there are sections for Integration runtimes, Microsoft Purview, Source control, Author, Security, and Workflow orchestration manager. The main area displays a list of 'Linked services' with four items: 'AzureDatabricks1' (Azure Databricks), 'AzureKeyVault' (Azure Key Vault), 'ls_adls_output' (Azure Data Lake Storage Gen2), and 'ls_http_input' (HTTP). To the right, a 'New linked service' dialog is open. It has fields for 'Name' (ls_synapse), 'Description' (empty), 'Connect via integration runtime' (set to 'AutoResolveIntegrationRuntime'), 'Version' (Recommended selected), 'Import from connection string' (unchecked), 'Account selection method' (From Azure subscription selected), 'Azure subscription' (Shrij-Wp (0b093c33-a116-4e12-aa43-073a1c622e60)), 'Server name' (health-synapse (Synapse workspace)), 'Database name' (shriji), 'SQL pool' (shriji), 'Authentication type' (SQL authentication), 'User name' (shriji), and 'Password' (Azure Key Vault selected). A 'Create' button is at the bottom.

Creating Dataset for Synapse

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar lists 'Factory Resources' including Pipelines, Datasets, Data flows, and Power Query. Under 'Datasets', 'ds_synapse_output' is selected. The main pane shows a pipeline named 'pipeline1' with a single step 'ds_synapse_output'. The 'Properties' panel on the right shows the dataset's properties: Name (ds_synapse_output) and Description (empty). The 'Connection' tab of the dataset's properties is selected, showing 'Linked service' (ls_synapse), 'Table' (dbo.heart_disease_data_internal), and a 'Test connection' button which is successful. Other tabs include 'Schema' and 'Parameters'.

Transferred Data in Synapse

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The left sidebar displays the 'Data' section with 'Workspace' selected, showing a tree view of 'SQL database' and 'shriji (SQL)' which contains 'Tables', 'External tables', 'External resources', 'Views', 'Programmability', 'Schemas', and 'Security'. The main area shows a 'Table_Creation' tab with a SQL script editor containing the following code:

```
1 SELECT * FROM dbo.heart_disease_data
```

The 'Properties' pane on the right shows the following details for 'SQL script 2':

- Name: SQL script 2
- Description: (empty)
- Type: sql script
- Size: 0 bytes
- Results settings per query:
 - First 5000 rows (default)
 - All rows

The 'Results' pane below shows the output of the query, displaying a table with the following data:

patient_state	patient_gender	general_health	physical_health...	mental_health...	last_checkup_ti...	physical_activit...	sleep_hours	had_heart_atta...	had_angina
Iowa	Female	Very good	0	0	Within past yea...	Yes	6	No	No
Iowa	Male	Good	0	0	Within past yea...	Yes	8	No	Yes
Iowa	Male	Fair	0	0	5 or more years...	No	6	No	No
Iowa	Male	Very good	0	2	Within past yea...	Yes	9	Yes	No
Iowa	Male	Excellent	0	0	5 or more years...	Yes	7	No	No
Iowa	Female	Good	0	0	Within past yea...	Yes	5	No	No

00:00:04 Retrieving query result.

Performing Different Analysis in Synapse

Microsoft Azure | Synapse Analytics > health-synapse

Develop SQL script 1 SQL script 2

```

21 -- Correlation Between Sleep Hours and BMI
22 SELECT sleep_hours, CEILING(AVG(bmi)) AS avg_bmi
23 FROM dbo.heart_disease_data
24 GROUP BY sleep_hours
25 ORDER BY sleep_hours;
26
27
28 --Impact of Physical Activities on Heart Attack Cases
29
30 SELECT physical_activities, COUNT(*) AS total_cases,
31 COUNT(CASE WHEN had_heart_attack = 'Yes' THEN 1 END) AS heart_attack_cases
32 FROM dbo.heart_disease_data
33 GROUP BY physical_activities
34 ORDER BY heart_attack_cases DESC;
35
36
37
38
39
40 --Frequency of Smoking Status by Gender.
41
42 SELECT patient_gender, smoker_status, COUNT(*) AS frequency
43 FROM dbo.heart_disease_data
44 GROUP BY patient_gender, smoker_status
45 ORDER BY frequency DESC;
46

```

Results Messages

View Table Chart Export results

patient_gender	smoker_status	frequency
Female	Never smoked	157042
Male	Never smoked	124375
Male	Former smoker	59531
Female	Former smoker	54243
Female	Current smoker - now smokes ev...	18082

00:00:03 Query executed successfully.

Microsoft Azure | Synapse Analytics > health-synapse

Develop SQL script 1 SQL script 2

```

21 -- Correlation Between Sleep Hours and BMI
22 SELECT sleep_hours, CEILING(AVG(bmi)) AS avg_bmi
23 FROM dbo.heart_disease_data
24 GROUP BY sleep_hours
25 ORDER BY sleep_hours;
26
27
28 --Impact of Physical Activities on Heart Attack Cases
29
30 SELECT physical_activities, COUNT(*) AS total_cases,
31 COUNT(CASE WHEN had_heart_attack = 'Yes' THEN 1 END) AS heart_attack_cases
32 FROM dbo.heart_disease_data
33 GROUP BY physical_activities
34 ORDER BY heart_attack_cases DESC;
35
36
37
38
39
40
41

```

Results Messages

View Table Chart Export results

physical_activities	total_cases	heart_attack_cases
Yes	337559	15285
No	107573	9823

00:00:02 Query executed successfully.

The screenshot shows the Microsoft Azure Synapse Analytics studio interface. On the left, there's a sidebar with icons for Home, Databases, Tables, and Functions. The main area has a 'Develop' tab selected. In the center, there are three tabs: 'SQL script 1' (selected), 'SQL script 2', and 'SQL script 3'. The 'SQL script 1' tab contains the following SQL code:

```
21 -- Correlation Between Sleep Hours and BMI
22 SELECT sleep_hours, CEILING(AVG(bmi)) AS avg_bmi
23 FROM dbo.heart_disease_data
24 GROUP BY sleep_hours
25 ORDER BY sleep_hours;
26
27
28
29 --Impact of Physical Activities on Heart Attack Cases
30
31 SELECT physical_activities, COUNT(*) AS total_cases,
32       COUNT(CASE WHEN had_heart_attack = 'Yes' THEN 1 END) AS heart_attack_cases
33 FROM dbo.heart_disease_data
34 GROUP BY physical_activities
35 ORDER BY heart_attack_cases DESC;
36
37
38
39
40 --Frequency of Smoking Status by Gender
41
42 SELECT patient_gender, smoker_status, COUNT(*) AS frequency
43 FROM dbo.heart_disease_data
44 GROUP BY patient_gender, smoker_status
45 ORDER BY frequency DESC;
```

To the right of the code editor is a 'Properties' panel with tabs for General, Related (0), Name (SQL script 1), Description, Type (sql script), Size (0 bytes), and Results settings per query (First 5000 rows (default)). Below the code editor is a 'Results' pane with a 'Table' view showing the following data:

patient_gender	smoker_status	frequency
Female	Never smoked	157042
Male	Never smoked	124375
Male	Former smoker	59531
Female	Former smoker	54243
Female	Current smoker - now smokes ev...	18082

At the bottom of the results pane, it says '00:00:03 Query executed successfully.'

The screenshot shows the Microsoft Azure Synapse Analytics development interface. The top navigation bar includes 'Microsoft Azure', 'Synapse Analytics', 'health-synapse', 'main branch', 'Validate all', 'Commit all', 'Publish', 'Search', and user information 'Shriji74@outlook.com DEFAULT DIRECTORY'. The left sidebar has a 'Develop' section with 'SQL scripts' (2), 'SQL script 1', and 'SQL script 2'. The main area displays a SQL script titled 'SQL script 1' with the following code:

```
34 GROUP BY physical_activities
35 ORDER BY heart_attack_cases DESC;
36
37
38
39
40 --Frequency of Smoking Status by Gender
41
42 SELECT patient_gender, smoker_status, COUNT(*) AS frequency
43 FROM dbo.heart_disease_data
44 GROUP BY patient_gender, smoker_status
45 ORDER BY frequency DESC;
46
47
48 --Count of Different Health Conditions by Age Category
49 SELECT age_category,
50 COUNT(CASE WHEN had_asthma = 'Yes' THEN 1 END) AS asthma_count,
51 COUNT(CASE WHEN had_skin_cancer = 'Yes' THEN 1 END) AS skin_cancer_count,
52 COUNT(CASE WHEN had_copd = 'Yes' THEN 1 END) AS copd_count,
53 COUNT(CASE WHEN had_depressive_disorder = 'Yes' THEN 1 END) AS depressive_disorder_count,
54 COUNT(CASE WHEN had_kidney_disease = 'Yes' THEN 1 END) AS kidney_disease_count,
55 FROM dbo.heart_disease_data
56 GROUP BY age_category
57 ORDER BY age_category;
```

The 'Results' tab is selected, showing a table with the following data:

age_category	asthma_count	skin_cancer_count	copd_count	depressive_disorder_count	kidney_disease...
Age 25 to 29	3955	98	437	5750	195
Age 30 to 34	4426	159	600	6718	315
Age 35 to 39	4601	327	834	7045	432
Age 40 to 44	4848	515	1104	7053	560
Age 45 to 49	4620	777	1408	6623	717

At the bottom, a message indicates '00:00:02 Query executed successfully.'

Creating Table in Dedicated SQL of those Analysis for Visualization

Microsoft Azure | Synapse Analytics > health-synapse

main branch | Validate all | Commit all | Publish

Search

Develop

SQL scripts

SQL script 1

SQL script 2

Create Table with ROUND_ROBIN Distribution

```
1 -- Create Table Average_BMI
2 CREATE TABLE Average_BMI
3 WITH(
4     DISTRIBUTION = ROUND_ROBIN
5 )
6 AS
7 SELECT patient_state, CEILING(AVG(bmi)) AS average_bmi
8 FROM dbo.heart_disease_data_internal
9 GROUP BY patient_state;
10
11
12
13
14
15 --Count of Heart Attack Cases by Age Category
16 CREATE TABLE Heart_Attack_Count
17 WITH(
18     DISTRIBUTION = ROUND_ROBIN
19 )
20 AS
21 SELECT age_category,COUNT(CASE WHEN had_heart_attack = 'Yes' THEN 1 END) AS heart_attack_count FROM dbo.heart_disease_data_internal
22 GROUP BY age_category
23
24
25 -- Distribution of Health Conditions by State
26 CREATE TABLE Health_Condition_State
27 WITH(

```

Properties

General Related (0)

Name * SQL script 1

Description

Type .sql script

Size 0 bytes

Results settings per query

First 5000 rows (default)

All rows

Results Messages

No results to show

Your query yielded no displayable results

Microsoft Azure | Synapse Analytics > health-synapse

Search

main branch > Validate all > Commit all > Publish

Data

SQL script 1 • SQL script 2

Run Undo Commit Query plan Connect to shrij... Use database shrij...

Filter resources by name

SQL database 1

shrij (SQL)

Tables

- dbo.Average_BMI
- dbo.Health_Condition_Age...
- dbo.Health_Condition_Status
- dbo.Heart_Attack_Count
- dbo.heart_disease_data
- dbo.heart_disease_data_int...
- dbo.PhysicalActivities_Heart...
- dbo.Sleep_Hours_BMI
- dbo.Smoke_Status

External tables

External resources

Views

Programmability

Schemas

Security

Properties

General Related (0)

Name * SQL script 1

Description

Type sql script

Size 0 bytes

Results settings per query ◉

First 5000 rows (default)

All rows

Results Messages

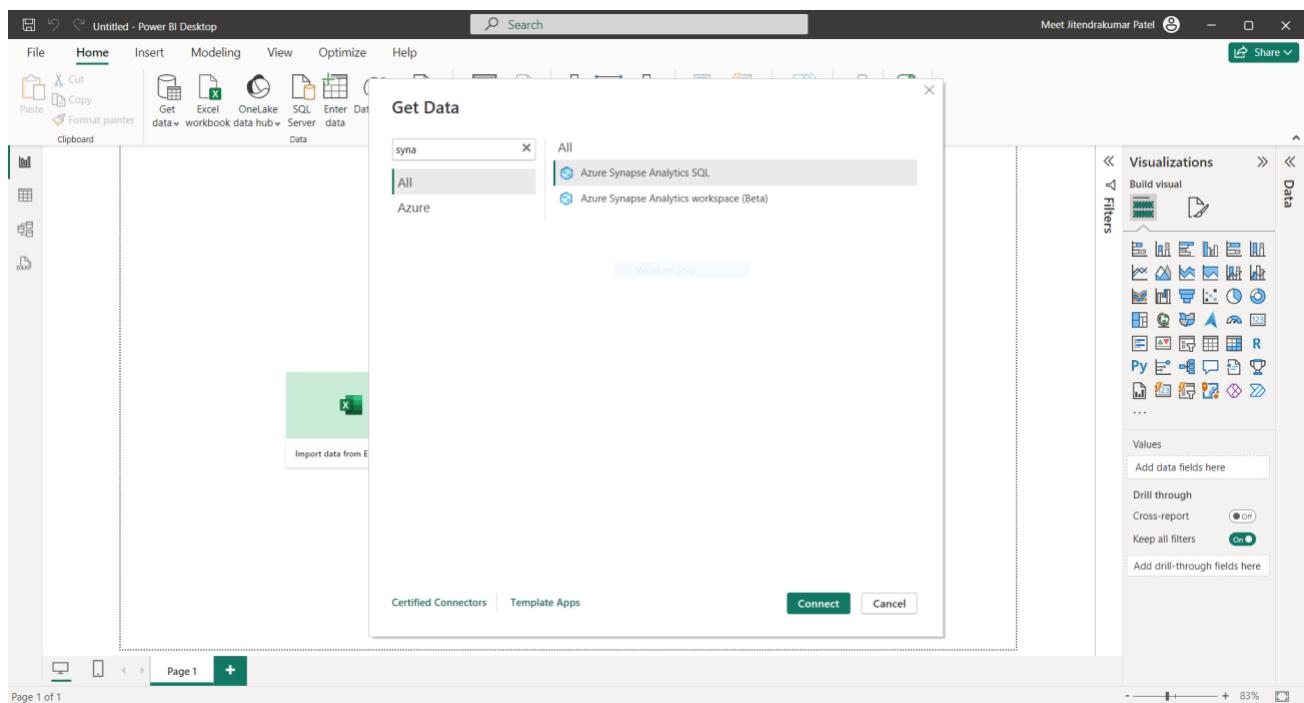
No results to show

Your query yielded no displayable results

2024-01-01 User executed successfully.

PowerBI

Getting Data from Azure Synapse Analytics

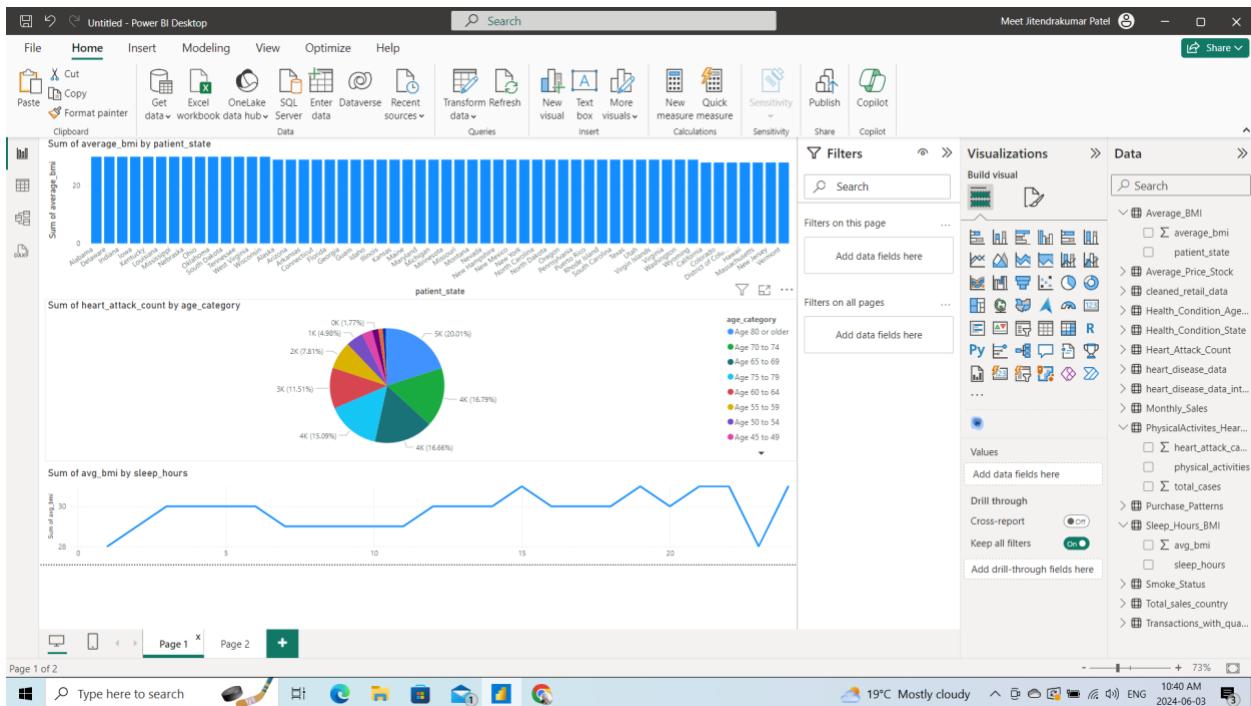


The screenshot shows the Power BI Desktop interface. The top navigation bar includes File, Home, Insert, Modeling, View, Optimize, and Help. The ribbon has sections for Paste, Cut, Copy, Get data, Excel, OneLake, and a Share button.

The main area features the Navigator pane on the left, which lists various data sources and tables. A table titled "Average_BMI" is displayed in the center, showing the average BMI for different US states. The Visualizations pane on the right offers options for building visualizations, filters, and drill-through reports.

patient_state	average_bmi
California	28
Louisiana	30
North Carolina	29
New Mexico	29
New Hampshire	29
Hawaii	28
District of Columbia	28
Minnesota	29
Pennsylvania	29
Ohio	30
Mississippi	30
Arizona	29
Missouri	29
Georgia	29
Oregon	29
Nebraska	30
Rhode Island	29
Wyoming	29
Indiana	30
Alaska	29
Alabama	30
Connecticut	29
Maryland	29
Virginia	29

Creating Different Visualization of the Table in Dedicated SQL pool



Untitled - Power BI Desktop

File Home Insert Modeling View Optimize Help

Clipboard Paste Cut Copy Format painter

Data Get data Excel OneLake SQL Server Enter data Recent sources Data Transform Refresh data New visual Text box insert More visuals Quick measure measure Calculations Sensitivity Share Publish Copilot

Queries

Sum of diabetes_count, Sum of heart_attack_count and Sum of stroke_count by patient_state

patient_state	Sum of diabetes_count	Sum of heart_attack_count	Sum of stroke_count
Washington	~3000	~1000	~1000
Ohio	~2800	~1000	~1000
Michigan	~2600	~1000	~1000
Texas	~2500	~1000	~1000
New York	~2400	~1000	~1000
Florida	~2300	~1000	~1000
Minnesota	~2200	~1000	~1000
Virginia	~2100	~1000	~1000
Indiana	~2000	~1000	~1000
Kansas	~1900	~1000	~1000
South Carolina	~1800	~1000	~1000
Azores	~1700	~1000	~1000
Georgia	~1600	~1000	~1000
Connecticut	~1500	~1000	~1000
Massachusetts	~1400	~1000	~1000
Michigan	~1300	~1000	~1000
Illinois	~1200	~1000	~1000
California	~1100	~1000	~1000
Massachusetts	~1000	~1000	~1000
Pennsylvania	~900	~1000	~1000
Utah	~800	~1000	~1000
Nebraska	~700	~1000	~1000
Arkansas	~600	~1000	~1000
Wyoming	~500	~1000	~1000
North Dakota	~400	~1000	~1000
South Dakota	~300	~1000	~1000
Louisiana	~200	~1000	~1000
Oklahoma	~100	~1000	~1000

Sum of heart_attack_cases and Sum of total_cases by physical_activities

physical_activities	Sum of heart_attack_cases	Sum of total_cases
No	~0.1M	~0.3M
Yes	~0.0M	~0.0M

Filters

Visualizations

Data

Build visual

Values

Add drill-through fields here

Drill through

Cross-report

Keep all filters

19°C Mostly cloudy 10:40 AM 2024-06-03

Type here to search

Conclusion

My project demonstrated the effectiveness of using data engineering techniques to analyze health related. By cleaning, transforming, and analyzing the data, the project provided valuable insights that can support healthcare decision-making, improve patient outcomes, and inform public health strategies. The use of Azure services, including Data Factory, Data Lake Storage, Databricks, Synapse Analytics, and Power BI, ensured the pipeline's scalability, efficiency, and effectiveness.

GitHub Link

[Github](#)