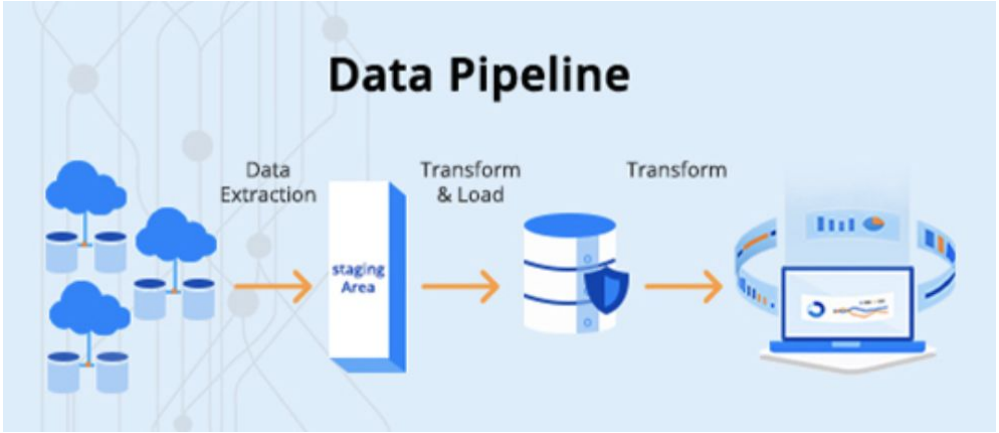
Comprehensive Data Pipeline Development for Healthcare Insurance Analytics

Capstone Project

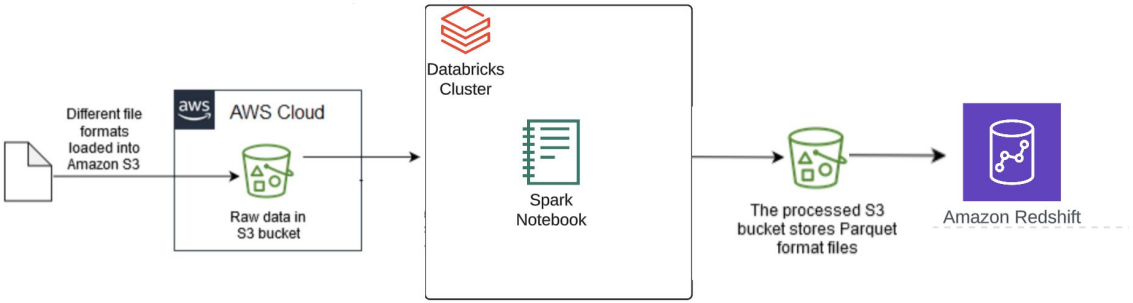


Presented By: Ekta Shrestha

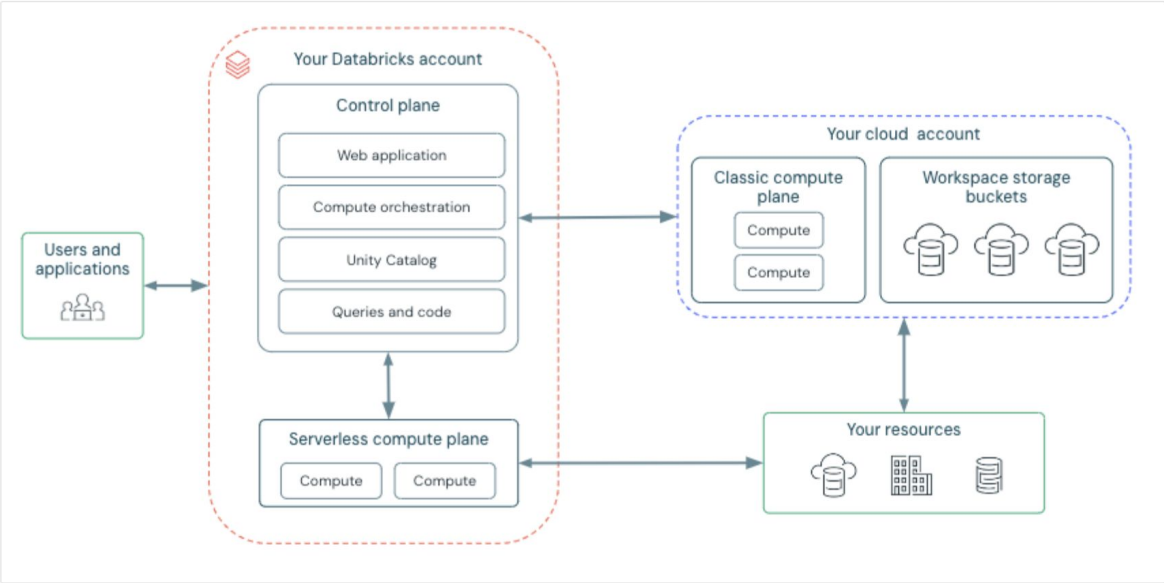
Requirement and Specification

- Enhance revenue and understand customer behaviors for Health Care insurance company.
- Develop data pipelines to:
  - Analyze customer behaviors.
  - Send customized offers.
  - Calculate and distribute royalties to past customers.
  - Develop business strategies to enhance revenue.

Solution Design Architecture Diagram



The following diagram describes the overall Databricks architecture.

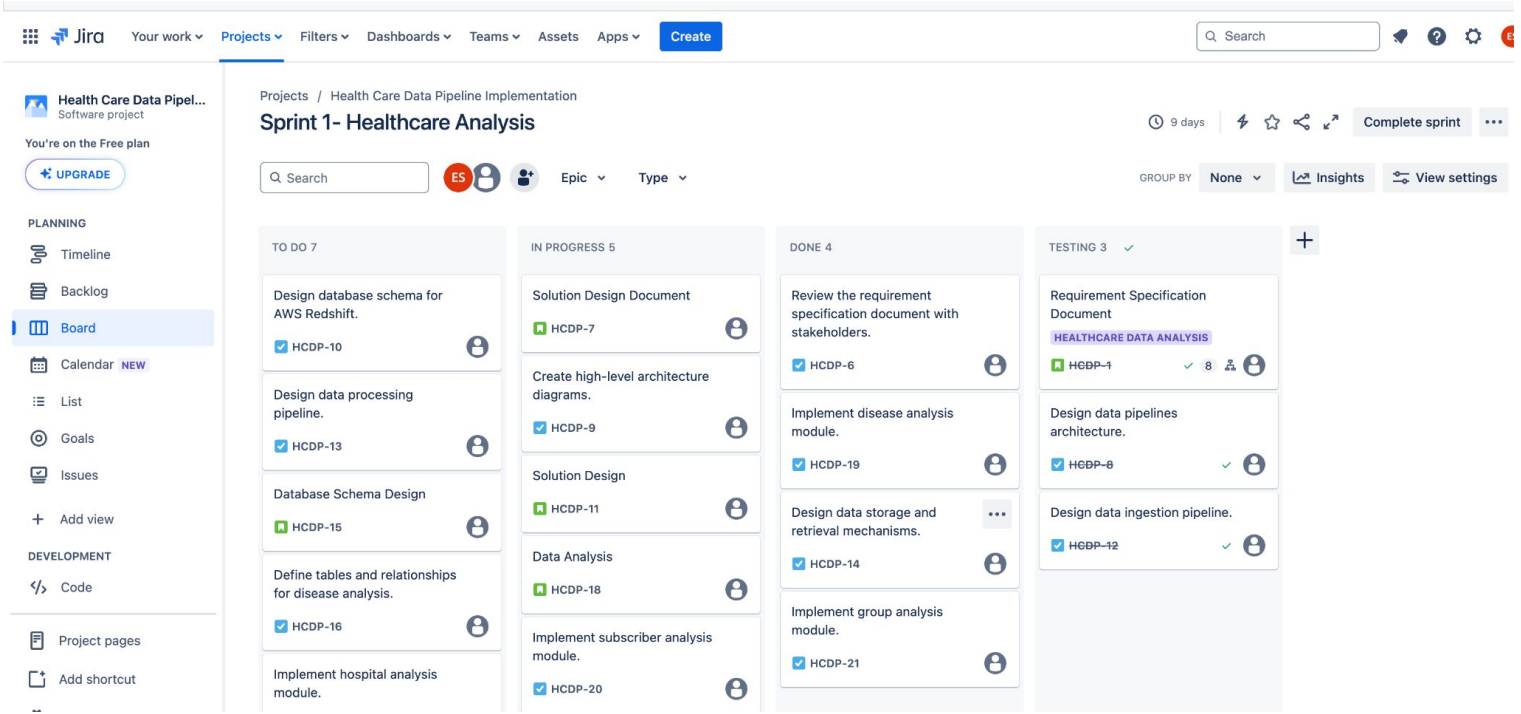


Environment

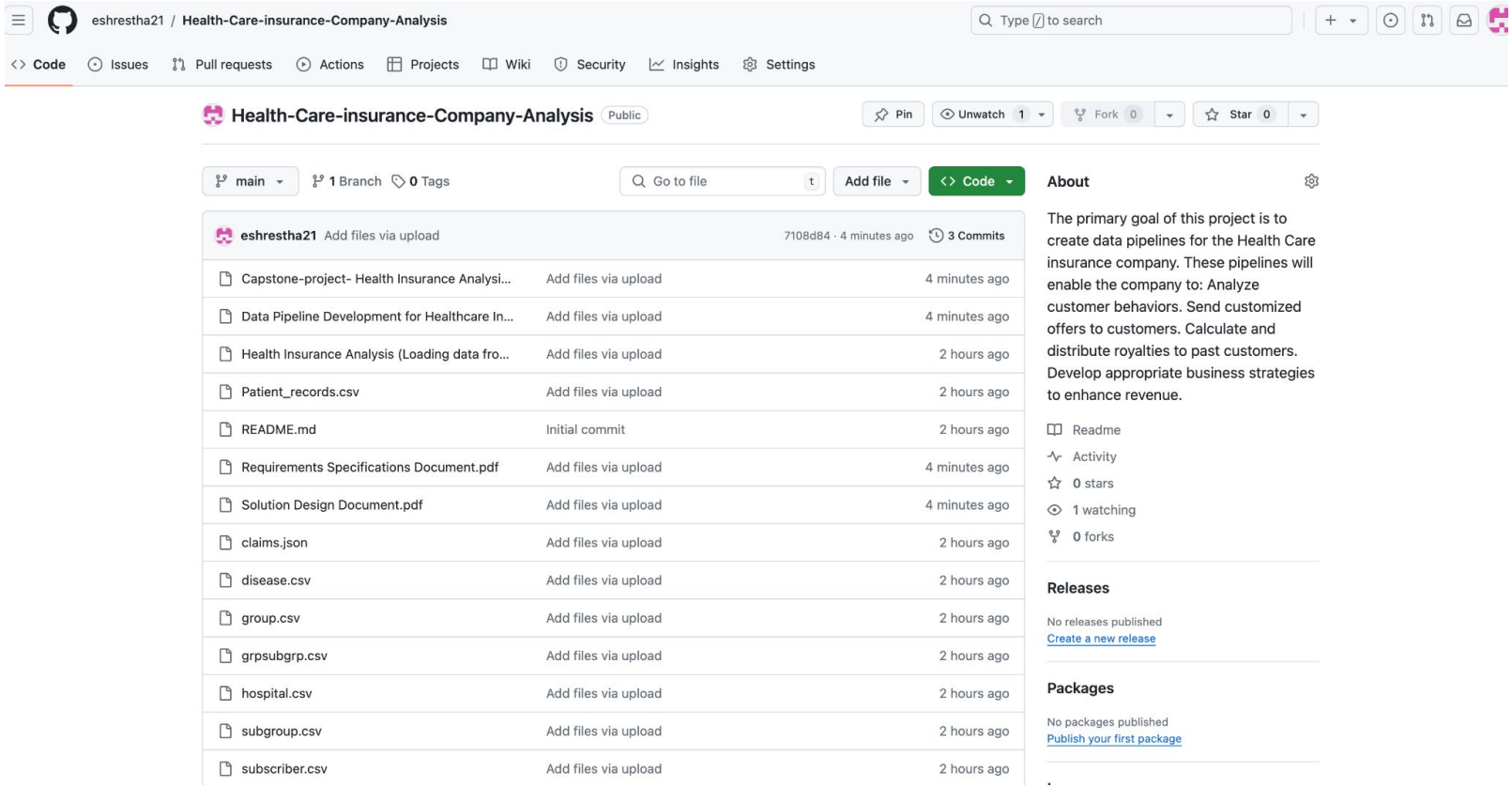
- **AWS S3:** Data storage for raw and processed data
- **AWS Redshift**: Data warehousing for analysis and reporting
- **Databricks**: Data processing and analysis platform
- **PySpark**: Distributed data processing framework
- **GitHub**: Version control and code repository
- **Jira**: Project management and tracking

Use Cases

1. **Disease Analysis**: Identify the disease with the maximum number of claims.
2. **Subscriber Analysis**: Find subscribers under the age of 30 who subscribe to any subgroup.
3. **Group Analysis**: Determine which group has the maximum number of subgroups.
4. **Hospital Analysis:** Identify the hospital that serves the most number of patients.
5. **Subgroup Analysis:** Find out which subgroup is subscribed to the most.
6. **Claims Rejection Analysis:** Calculate the total number of claims that were rejected.
7. **Claims Origin Analysis:** Determine the city from which most claims are coming.
8. **Policy Type Analysis:** Identify whether subscribers mostly subscribe to government or private policies.
9. **Premium Analysis:** Calculate the average monthly premium paid by subscribers to the insurance company.
10. **Profitability Analysis:** Find out which group is most profitable.
11. **Pediatric Cancer Patients:** List all patients below age 18 who are admitted for cancer.
12. **High-Value Cashless Insurance Patients:** List patients who have cashless insurance and total charges greater than or equal to Rs. 50,000.
13. **Female Knee Surgery Patients:** List female patients over the age of 40 that have undergone knee surgery in the past year.


Jira
Your work
Projects
Filters
Dashboards
Teams
Assets
Apps
Create
Search
Health Care Data Pipel...
Software project
You're on the Free plan
UPGRADE
PLANNING
Timeline
Backlog
Board
Calendar NEW
List
Goals
Issues
Add view
DEVELOPMENT
Code
Project pages
Add shortcut
Projects / Health Care Data Pipeline Implementation
Sprint 1- Healthcare Analysis
9 days
Complete sprint
Search
ES
Epic
Type
GROUP BY None
Insights
View settings
TO DO 7
Design database schema for AWS Redshift.
HCDP-10
Design data processing pipeline.
HCDP-13
Database Schema Design
HCDP-15
Define tables and relationships for disease analysis.
HCDP-16
Implement hospital analysis module.
IN PROGRESS 5
Solution Design Document
HCDP-7
Create high-level architecture diagrams.
HCDP-9
Solution Design
HCDP-11
Data Analysis
HCDP-18
Implement subscriber analysis module.
HCDP-20
DONE 4
Review the requirement specification document with stakeholders.
HCDP-6
Implement disease analysis module.
HCDP-19
Design data storage and retrieval mechanisms.
HCDP-14
Implement group analysis module.
HCDP-21
TESTING 3
Requirement Specification Document
HEALTHCARE DATA ANALYSIS
HCDP-1
8
Design data pipelines architecture.
HCDP-8
Design data ingestion pipeline.
HCDP-12

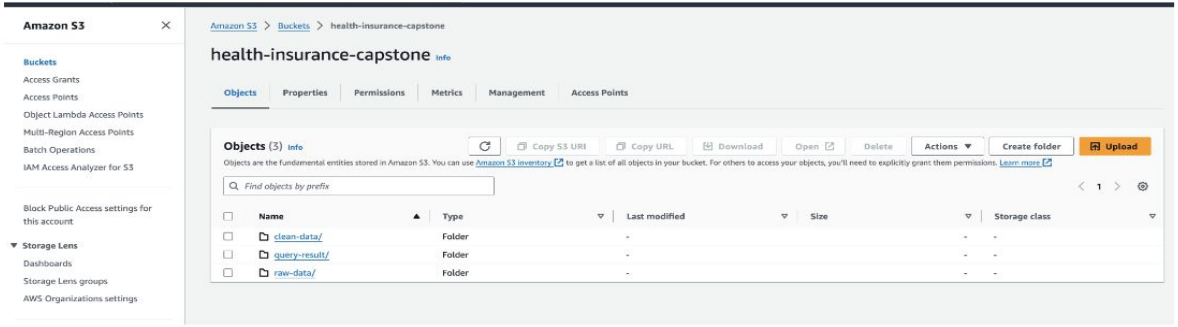
Github

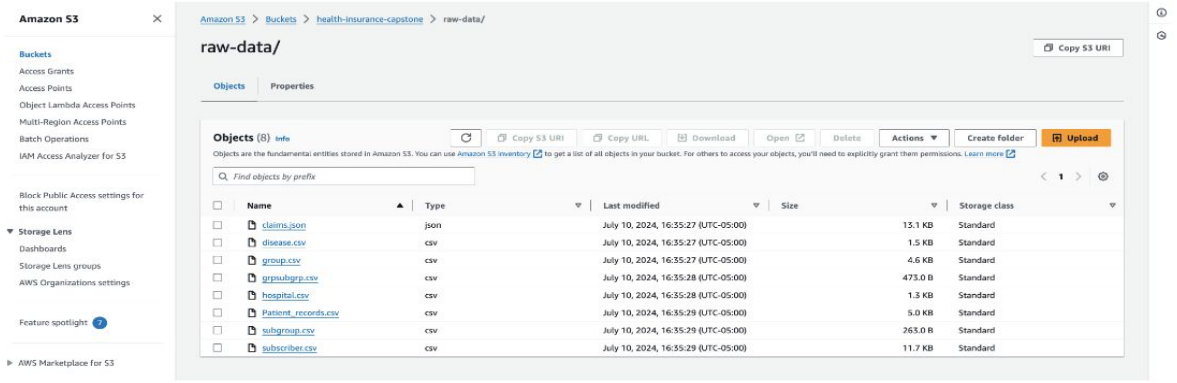

eshrestha21 / Health-Care-insurance-Company-Analysis
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Health-Care-insurance-Company-Analysis
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
t
Add file
Code
eshrestha21 Add files via upload
7108d84 · 4 minutes ago
3 Commits
Capstone-project- Health Insurance Analysi... Add files via upload 4 minutes ago
Data Pipeline Development for Healthcare In... Add files via upload 4 minutes ago
Health Insurance Analysis (Loading data fro... Add files via upload 2 hours ago
Patient_records.csv Add files via upload 2 hours ago
README.md Initial commit 2 hours ago
Requirements Specifications Document.pdf Add files via upload 4 minutes ago
Solution Design Document.pdf Add files via upload 4 minutes ago
claims.json Add files via upload 2 hours ago
disease.csv Add files via upload 2 hours ago
group.csv Add files via upload 2 hours ago
grpsubgrp.csv Add files via upload 2 hours ago
hospital.csv Add files via upload 2 hours ago
subgroup.csv Add files via upload 2 hours ago
subscriber.csv Add files via upload 2 hours ago
About
The primary goal of this project is to create data pipelines for the Health Care insurance company. These pipelines will enable the company to: Analyze customer behaviors. Send customized offers to customers. Calculate and distribute royalties to past customers. Develop appropriate business strategies to enhance revenue.
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

Dataset Creation

Environment: Databricks, PySpark, AWS S3

Data stored in AWS S3





Accessing the AWS Bucket

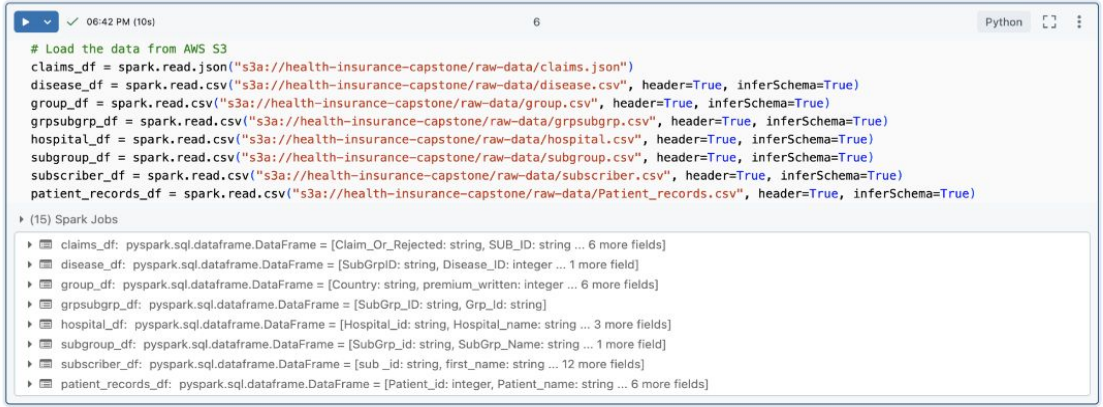
02:21 PM (<1s) 3

```
#Import libraries
from pyspark.sql import SparkSession, Row
from pyspark.sql.functions import col, count, avg, when, year,datediff, current_date, avg, sum, count
from pyspark.sql.types import IntegerType

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Health Insurance Analysis") \
    .getOrCreate()
```

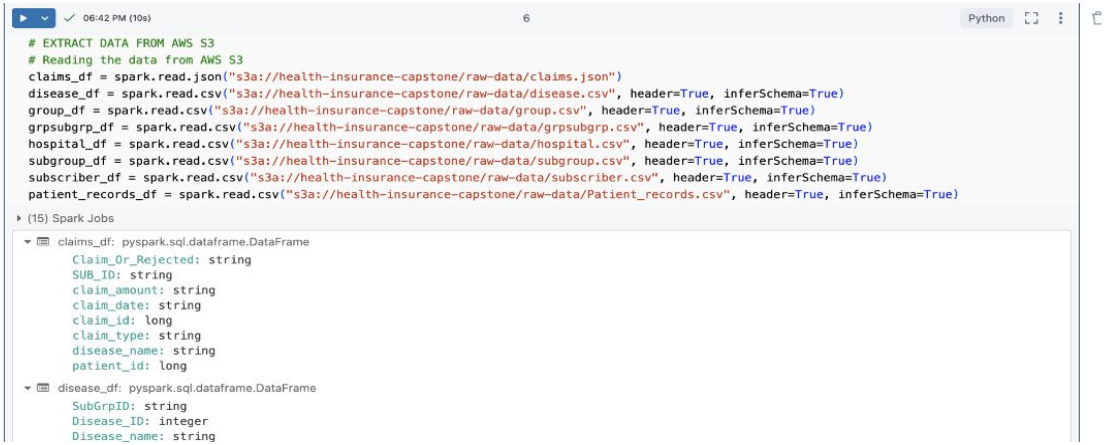
02:21 PM (<1s) 4

```
# Accessing the AWS Bucket
access_key = 'XXXXXXXXXXx'
secret_key = 'XXXXXX+xIP9o/I'
sc._jsc.hadoopConfiguration().set("fs.s3a.access.key", access_key)
sc._jsc.hadoopConfiguration().set("fs.s3a.secret.key", secret_key)

# If you are using Auto Loader file notification mode to load files, provide the AWS Region ID.
aws_region = "us-east-2"
sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint", "s3." + aws_region + ".amazonaws.com")
```

#ExtractLoad Data in Databricks from AWS S3



```
06:42 PM (10s)                                                    6                    Python

# Load the data from AWS S3
claims_df = spark.read.json("s3a://health-insurance-capstone/raw-data/claims.json")
disease_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/disease.csv", header=True, inferSchema=True)
group_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/group.csv", header=True, inferSchema=True)
grpsubgrp_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/grpsubgrp.csv", header=True, inferSchema=True)
hospital_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/hospital.csv", header=True, inferSchema=True)
subgroup_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/subgroup.csv", header=True, inferSchema=True)
subscriber_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/subscriber.csv", header=True, inferSchema=True)
patient_records_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/Patient_records.csv", header=True, inferSchema=True)

▸ (15) Spark Jobs
▸ claims_df: pyspark.sql.dataframe.DataFrame = [Claim_Or_Rejected: string, SUB_ID: string ... 6 more fields]
▸ disease_df: pyspark.sql.dataframe.DataFrame = [SubGrpID: string, Disease_ID: integer ... 1 more field]
▸ group_df: pyspark.sql.dataframe.DataFrame = [Country: string, premium_written: integer ... 6 more fields]
▸ grpsubgrp_df: pyspark.sql.dataframe.DataFrame = [SubGrp_ID: string, Grp_Id: string]
▸ hospital_df: pyspark.sql.dataframe.DataFrame = [Hospital_id: string, Hospital_name: string ... 3 more fields]
▸ subgroup_df: pyspark.sql.dataframe.DataFrame = [SubGrp_id: string, SubGrp_Name: string ... 1 more field]
▸ subscriber_df: pyspark.sql.dataframe.DataFrame = [sub _id: string, first_name: string ... 12 more fields]
▸ patient_records_df: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 6 more fields]
```



```
06:42 PM (10s)                                                    6                    Python

# EXTRACT DATA FROM AWS S3
# Reading the data from AWS S3
claims_df = spark.read.json("s3a://health-insurance-capstone/raw-data/claims.json")
disease_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/disease.csv", header=True, inferSchema=True)
group_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/group.csv", header=True, inferSchema=True)
grpsubgrp_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/grpsubgrp.csv", header=True, inferSchema=True)
hospital_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/hospital.csv", header=True, inferSchema=True)
subgroup_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/subgroup.csv", header=True, inferSchema=True)
subscriber_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/subscriber.csv", header=True, inferSchema=True)
patient_records_df = spark.read.csv("s3a://health-insurance-capstone/raw-data/Patient_records.csv", header=True, inferSchema=True)

▸ (15) Spark Jobs
▾ claims_df: pyspark.sql.dataframe.DataFrame
    Claim_Or_Rejected: string
    SUB_ID: string
    claim_amount: string
    claim_date: string
    claim_id: long
    claim_type: string
    disease_name: string
    patient_id: long
▾ disease_df: pyspark.sql.dataframe.DataFrame
    SubGrpID: string
    Disease_ID: integer
    Disease_name: string
```

Just now (1s) 8 Python

```
# Print schema and display data for each DataFrame
print("Schema of claims_df:")
claims_df.printSchema()
claims_df.show()
```

(1) Spark Jobs

```
Schema of claims_df:
root
 |-- Claim_Or_Rejected: string (nullable = true)
 |-- SUB_ID: string (nullable = true)
 |-- claim_amount: string (nullable = true)
 |-- claim_date: string (nullable = true)
 |-- claim_id: long (nullable = true)
 |-- claim_type: string (nullable = true)
 |-- disease_name: string (nullable = true)
 |-- patient_id: long (nullable = true)

+----------------+----------+------------+----------+--------+---------------+---------------+----------+
|Claim_Or_Rejected|    SUB_ID|claim_amount|claim_date|claim_id|     claim_type|   disease_name|patient_id|
+----------------+----------+------------+----------+--------+---------------+---------------+----------+
|               N| SUBID1000|       79874|1949-03-14|       0| claims of value|   Galactosemia|    187158|
|             NaN|SUBID10001|      151142|1970-03-16|       1|claims of policy| Bladder cancer|    112766|
|             NaN|SUBID10002|       59924|2008-02-03|       2| claims of value|  Kidney cancer|    199252|
|             NaN|SUBID10003|      143120|1995-02-08|       3|  claims of fact|        Suicide|    133424|
|               Y|SUBID10004|      168634|1967-05-23|       4| claims of value|   Food allergy|    172579|
|             NaN|SUBID10005|       64840|1991-10-04|       5|claims of policy|       Whiplash|    171320|
|               N| SUBID1006|       26800|1991-03-26|       6|  claims of fact|     Sunbathing|    107794|
```

Just now (<1s) 8

```
# Print schema and display data for each DataFrame
print("Schema of group_df:")
group_df.printSchema()
group_df.show()
```

(1) Spark Jobs

```
Schema of group_df:
root
 |-- Country: string (nullable = true)
 |-- premium_written: integer (nullable = true)
 |-- zipcode: integer (nullable = true)
 |-- Grp_Id: string (nullable = true)
 |-- Grp_Name: string (nullable = true)
 |-- Grp_Type: string (nullable = true)
 |-- city: string (nullable = true)
 |-- year: integer (nullable = true)

+-------+---------------+-------+------+--------------------+--------+---------+----+
|Country|premium_written|zipcode|Grp_Id|            Grp_Name|Grp_Type|     city|year|
+-------+---------------+-------+------+--------------------+--------+---------+----+
|  India|          72000| 482018|GRP101|Life Insurance Co...|   Govt.|   Mumbai|1956|
|  India|          45000| 482049|GRP102|HDFC Standard Lif...| Private|   Mumbai|2000|
|  India|          64000| 482030|GRP103|Max Life Insuranc...| Private|    Delhi|2000|
|  India|          59000| 482028|GRP104|ICICI Prudential ...| Private|   Mumbai|2000|
|  India|          37000| 482014|GRP105|Kotak Mahindra Li...| Private|   Mumbai|2001|
|  India|          89000| 482011|GRP106|Aditya Birla Sun ...| Private|   Mumbai|2000|
|  India|          70000| 482006|GRP107|TATA AIG Life Ins...| Private|   Mumbai|2001|
```

Data Cleaning

- Process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset
- Cleaning Activity
  - First check if there are null values in dataset
  - Count the total Null values for each column
  - And then replace the null values for specific columns by NA
  - Check the If three are duplicates records
  - If there are duplicates then drop duplicates

```python
# TRANSFORM DATA

# Function to clean a DataFrame
def clean_data(df, df_name):
    # 1. Replace empty strings with None (null)
    df = df.replace('', None)

    # 2. Check for null values and count them for each column
    null_counts = df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns])
    null_counts.show()

    # 3. Replace remaining null values with 'NA' for specific columns (customize as needed)
    columns_to_replace = df.columns
    df = df.fillna("NA", subset=columns_to_replace)

    # 4. Check for duplicates and drop duplicates
    initial_count = df.count()
    df = df.dropDuplicates()
    final_count = df.count()

    print(f"Number of duplicates dropped in {df_name}: {initial_count - final_count}")

return df
```

07:01 PM (14s) 12

```
# Clean each DataFrame using user defined function clean_data()
claims_df_cleaned = clean_data(claims_df, "claims_df")
disease_df_cleaned = clean_data(disease_df, "disease_df")
group_df_cleaned = clean_data(group_df, "group_df")
grpsubgrp_df_cleaned = clean_data(grpsubgrp_df, "grpsubgrp_df")
hospital_df_cleaned = clean_data(hospital_df, "hospital_df")
subgroup_df_cleaned = clean_data(subgroup_df, "subgroup_df")
subscriber_df_cleaned = clean_data(subscriber_df, "subscriber_df")
patient_records_df_cleaned = clean_data(patient_records_df, "patient_records_df")
```

(56) Spark Jobs

- claims_df_cleaned: pyspark.sql.dataframe.DataFrame = [Claim_Or_Rejected: string, SUB_ID: string ... 6 more fields]
- disease_df_cleaned: pyspark.sql.dataframe.DataFrame = [SubGrpID: string, Disease_ID: integer ... 1 more field]
- group_df_cleaned: pyspark.sql.dataframe.DataFrame = [Country: string, premium_written: integer ... 6 more fields]
- grpsubgrp_df_cleaned: pyspark.sql.dataframe.DataFrame = [SubGrp_ID: string, Grp_Id: string]
- hospital_df_cleaned: pyspark.sql.dataframe.DataFrame = [Hospital_id: string, Hospital_name: string ... 3 more fields]
- subgroup_df_cleaned: pyspark.sql.dataframe.DataFrame = [SubGrp_id: string, SubGrp_Name: string ... 1 more field]
- subscriber_df_cleaned: pyspark.sql.dataframe.DataFrame = [sub _id: string, first_name: string ... 12 more fields]
- patient_records_df_cleaned: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 6 more fields]

```
+-----------------+------+-----------+----------+--------+----------+------------+----------+
|Claim_Or_Rejected|SUB_ID|claim_amount|claim_date|claim_id|claim_type|disease_name|patient_id|
+-----------------+------+-----------+----------+--------+----------+------------+----------+
|                0|     0|          0|         0|       0|         0|           0|         0|
+-----------------+------+-----------+----------+--------+----------+------------+----------+

Number of duplicates dropped in claims_df: 0
+--------+-----------+------------+
|SubGrpID| Disease_ID|Disease_name|
+--------+-----------+------------+
|       0|          0|           0|
+--------+-----------+------------+
```

```
Number of duplicates dropped in disease_df: 0
+-------+---------------+-------+------+--------+--------+----+----+
|Country|premium_written|zipcode|Grp_Id|Grp_Name|Grp_Type|city|year|
+-------+---------------+-------+------+--------+--------+----+----+
|      0|              0|      0|     0|       0|       0|   0|   0|
+-------+---------------+-------+------+--------+--------+----+----+

Number of duplicates dropped in group_df: 0
+---------+------+
|SubGrp_ID|Grp_Id|
+---------+------+
|        0|     0|
+---------+------+

Number of duplicates dropped in grpsubgrp_df: 0
+-----------+-------------+----+-----+-------+
|Hospital_id|Hospital_name|city|state|country|
+-----------+-------------+----+-----+-------+
|          0|            0|   0|    0|      0|
+-----------+-------------+----+-----+-------+
```

▸ subgroup_df_cleaned: pyspark.sql.dataframe.DataFrame = [SubGrp_id: string, SubGrp_Name: string ... 1 more field]

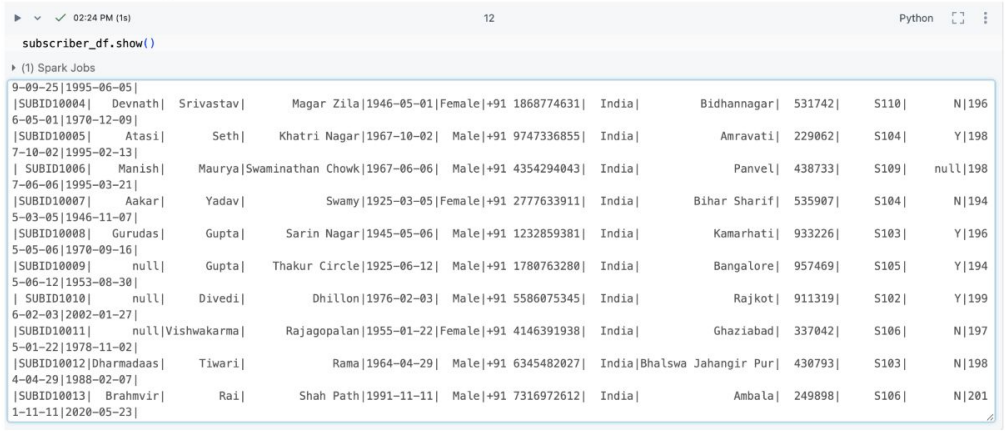
▸ subscriber_df_cleaned: pyspark.sql.dataframe.DataFrame = [sub _id: string, first_name: string ... 12 more fields]

▸ patient_records_df_cleaned: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 6 more fields]

```
+---------+-----------+---------------+
|SubGrp_id|SubGrp_Name|Monthly_Premium|
+---------+-----------+---------------+
|        0|          0|              0|
+---------+-----------+---------------+

Number of duplicates dropped in subgroup_df: 0
+-------+----------+---------+------+----------+------+-----+-------+----+--------+---------+--------+--------+---------+
|sub _id|first_name|last_name|Street|Birth_date|Gender|Phone|Country|City|Zip Code|Subgrp_id|Elig_ind|eff_date|term_date|
+-------+----------+---------+------+----------+------+-----+-------+----+--------+---------+--------+--------+---------+
|      0|        27|        0|     0|         0|     0|    3|      0|   0|       0|        2|       4|       0|        0|
+-------+----------+---------+------+----------+------+-----+-------+----+--------+---------+--------+--------+---------+

Number of duplicates dropped in subscriber_df: 0
+----------+------------+--------------+-----------------+-------------+------------+----+-----------+
|Patient_id|Patient_name|patient_gender|patient_birth_date|patient_phone|disease_name|city|hospital_id|
+----------+------------+--------------+-----------------+-------------+------------+----+-----------+
|         0|          17|             0|                0|            2|           0|   0|          0|
+----------+------------+--------------+-----------------+-------------+------------+----+-----------+

Number of duplicates dropped in patient_records_df: 0
```

Table: Subscriber

Before Cleaning



After Cleaning

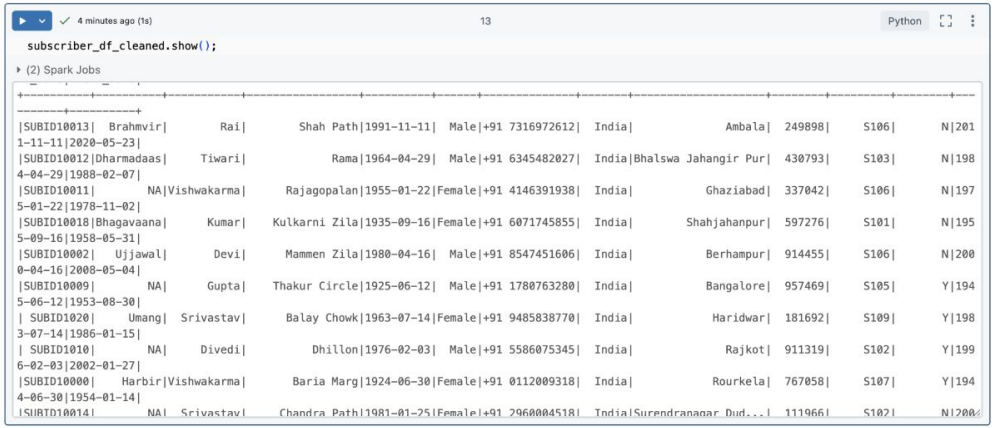


Table: Patient

Before Cleaning

02:37 PM (1s) 14

```
patient_records_df.show()
```

(1) Spark Jobs

```
|    187158|      Harbir|        Female|       1924-06-30|+91 0112009318|    Galactosemia|           Rourkela|      H1001|
|    112766|    Brahmdev|        Female|       1948-12-20|+91 1727749552|  Bladder cancer|       Tiruvottiyur|      H1016|
|    199252|     Ujjawal|          Male|       1980-04-16|+91 8547451606|   Kidney cancer|          Berhampur|      H1009|
|    133424|     Ballari|        Female|       1969-09-25|+91 0106026841|         Suicide|       Bihar Sharif|      H1017|
|    172579|     Devnath|        Female|       1946-05-01|+91 1868774631|    Food allergy|        Bidhannagar|      H1019|
|    171320|       Atasi|          Male|       1967-10-02|+91 9747336855|        Whiplash|           Amravati|      H1013|
|    107794|      Manish|          Male|       1967-06-06|+91 4354294043|      Sunbathing|             Panvel|      H1004|
|    130339|       Aakar|        Female|       1925-03-05|+91 2777633911|Drug consumption|       Bihar Sharif|      H1000|
|    110377|     Gurudas|          Male|       1945-05-06|+91 1232859381|          Dengue|          Kamarhati|      H1001|
|    149367|        null|          Male|       1925-06-12|+91 1780763280|    Head banging|          Bangalore|      H1013|
|    156168|        null|          Male|       1976-02-03|+91 5586075345| Fanconi anaemia|             Rajkot|      H1004|
|    114241|        null|        Female|       1955-01-22|+91 4146391938|   Breast cancer|          Ghaziabad|      H1015|
|    146382|  Dharmadaas|          Male|       1964-04-29|+91 6345482027|         Anthrax|Bhalswa Jahangir Pur|      H1019|
|    132748|    Brahmvir|          Male|       1991-11-11|+91 7316972612| Cystic fibrosis|             Ambala|      H1018|
|    167340|        null|        Female|       1981-01-25|+91 2960004518|    Galactosemia|Surendranagar Dud...|      H1003|
|    135184|     Bhagvan|        Female|       1966-07-24|+91 0297693485|          Dengue|          Bhimavaram|      H1018|
|    179662|   Amritkala|        Female|       1933-11-20|+91 0537157280|        Smallpox|              Meerut|      H1018|
|    184479|      Bandhu|          Male|       1996-10-15|+91 0695289163|  Pollen allergy|           Chinsurah|      H1010|
|    156988|  Bhagavaana|        Female|       1935-09-16|+91 6071745855|   Breast cancer|       Shahjahanpur|      H1012|
|    132870|        null|        Female|       1924-11-09|+91 8906694405|        Glaucoma|           Jabalpur|      H1017|
+----------+------------+--------------+-----------------+--------------+----------------+--------------------+-----------+
```

After Cleaning

02:37 PM (1s) 15

```
patient_records_df_cleaned.show()
```

(2) Spark Jobs

```
|    132748|    Brahmvir|          Male|       1991-11-11|+91 7316972612| Cystic fibrosis|             Ambala|      H1018|
|    172579|     Devnath|        Female|       1946-05-01|+91 1868774631|    Food allergy|        Bidhannagar|      H1019|
|    146382|  Dharmadaas|          Male|       1964-04-29|+91 6345482027|         Anthrax|Bhalswa Jahangir Pur|      H1019|
|    114241|          NA|        Female|       1955-01-22|+91 4146391938|   Breast cancer|          Ghaziabad|      H1015|
|    148137|       Umang|        Female|       1963-07-14|+91 9485838770|     Pet allergy|           Haridwar|      H1002|
|    171320|       Atasi|          Male|       1967-10-02|+91 9747336855|        Whiplash|           Amravati|      H1013|
|    187158|      Harbir|        Female|       1924-06-30|+91 0112009318|    Galactosemia|           Rourkela|      H1001|
|    199252|     Ujjawal|          Male|       1980-04-16|+91 8547451606|   Kidney cancer|          Berhampur|      H1009|
|    149367|          NA|          Male|       1925-06-12|+91 1780763280|    Head banging|          Bangalore|      H1013|
|    133424|     Ballari|        Female|       1969-09-25|+91 0106026841|         Suicide|       Bihar Sharif|      H1017|
|    130339|       Aakar|        Female|       1925-03-05|+91 2777633911|Drug consumption|       Bihar Sharif|      H1000|
|    156988|  Bhagavaana|        Female|       1935-09-16|+91 6071745855|   Breast cancer|       Shahjahanpur|      H1012|
|    112766|    Brahmdev|        Female|       1948-12-20|+91 1727749552|  Bladder cancer|       Tiruvottiyur|      H1016|
|    107794|      Manish|          Male|       1967-06-06|+91 4354294043|      Sunbathing|             Panvel|      H1004|
|    110377|     Gurudas|          Male|       1945-05-06|+91 1232859381|          Dengue|          Kamarhati|      H1001|
|    132870|          NA|        Female|       1924-11-09|+91 8906694405|        Glaucoma|           Jabalpur|      H1017|
|    179662|   Amritkala|        Female|       1933-11-20|+91 0537157280|        Smallpox|              Meerut|      H1018|
|    184479|      Bandhu|          Male|       1996-10-15|+91 0695289163|  Pollen allergy|           Chinsurah|      H1010|
+----------+------------+--------------+-----------------+--------------+----------------+--------------------+-----------+
only showing top 20 rows
```

Writing the cleaned data to AWS S3

- After cleaning and analysis, processed/cleaned data which is exported back to our AWS S3 bucket, creating a new 'cleaned-data' folder.
- This helps to ensure that we maintain a clear separation between raw and processed data, and provides a clean dataset for future use or additional analytics.
- Environment: Databricks, PySpark, AWS S3

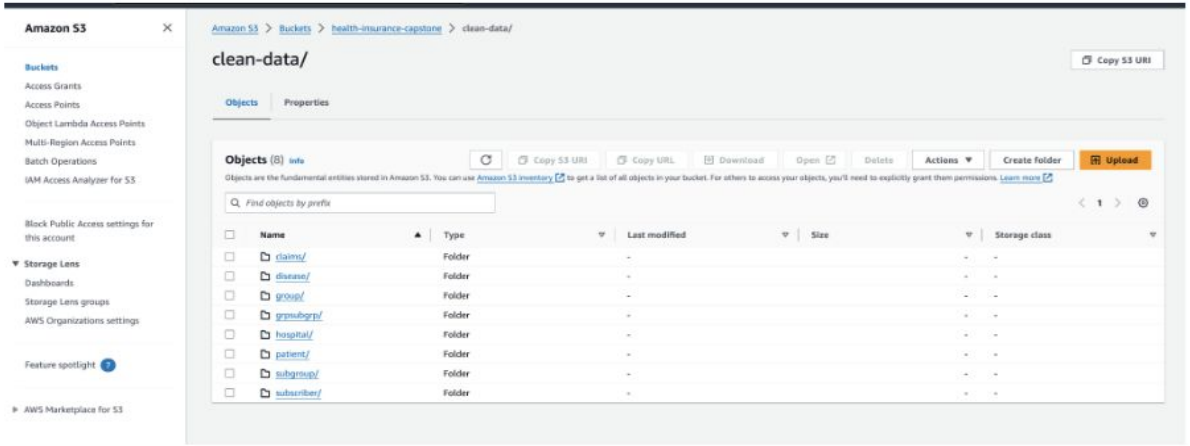
```
# WRITING DATA
```

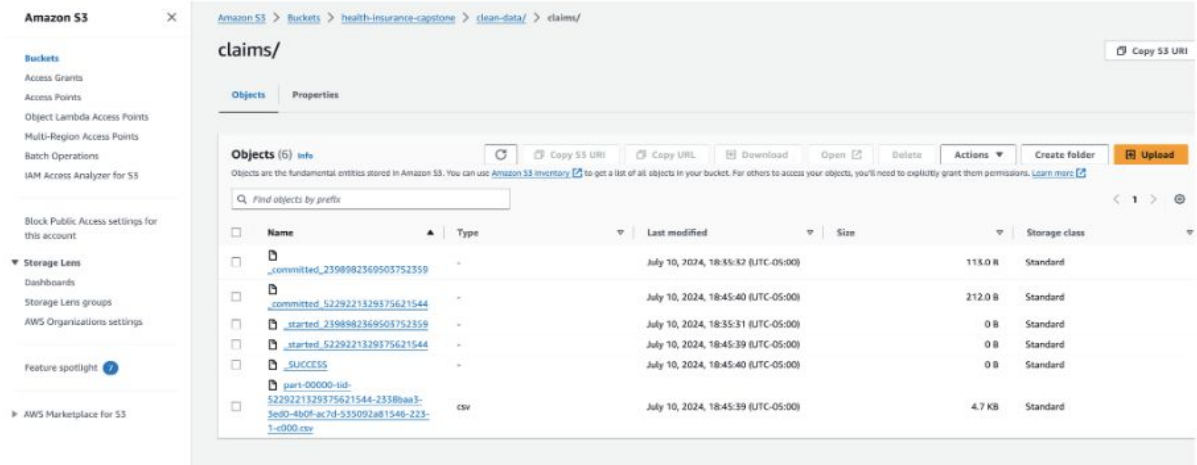
1 minute ago (27s) 14 Python

```
# # Writing the cleaned data to AWS S3
claims_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/claims")
disease_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/disease")
group_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/group")
grpsubgrp_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/grpsubgrp")
hospital_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/hospital")
subgroup_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/subgroup")
subscriber_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/subscriber")
patient_records_df_cleaned.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/clean-data/patient")
```

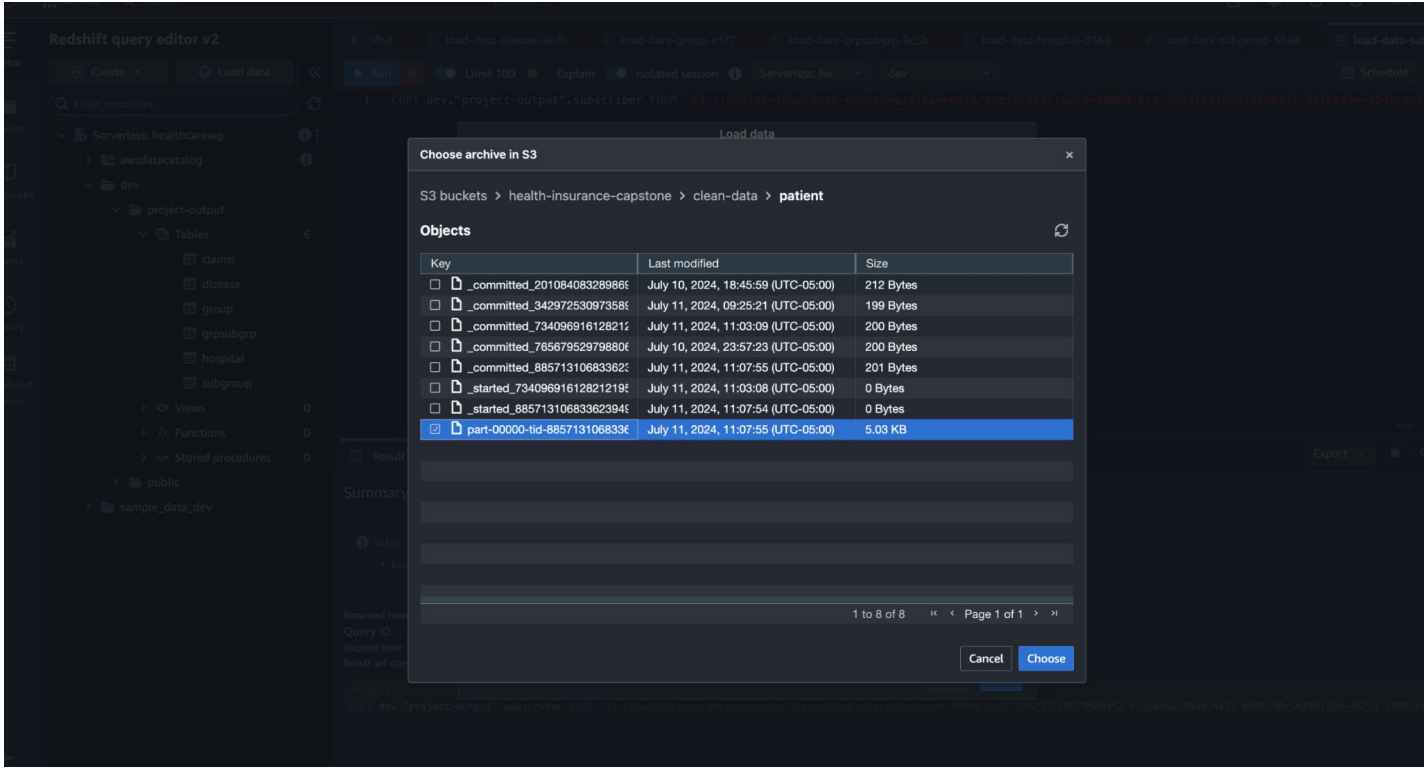
▾ (14) Spark Jobs

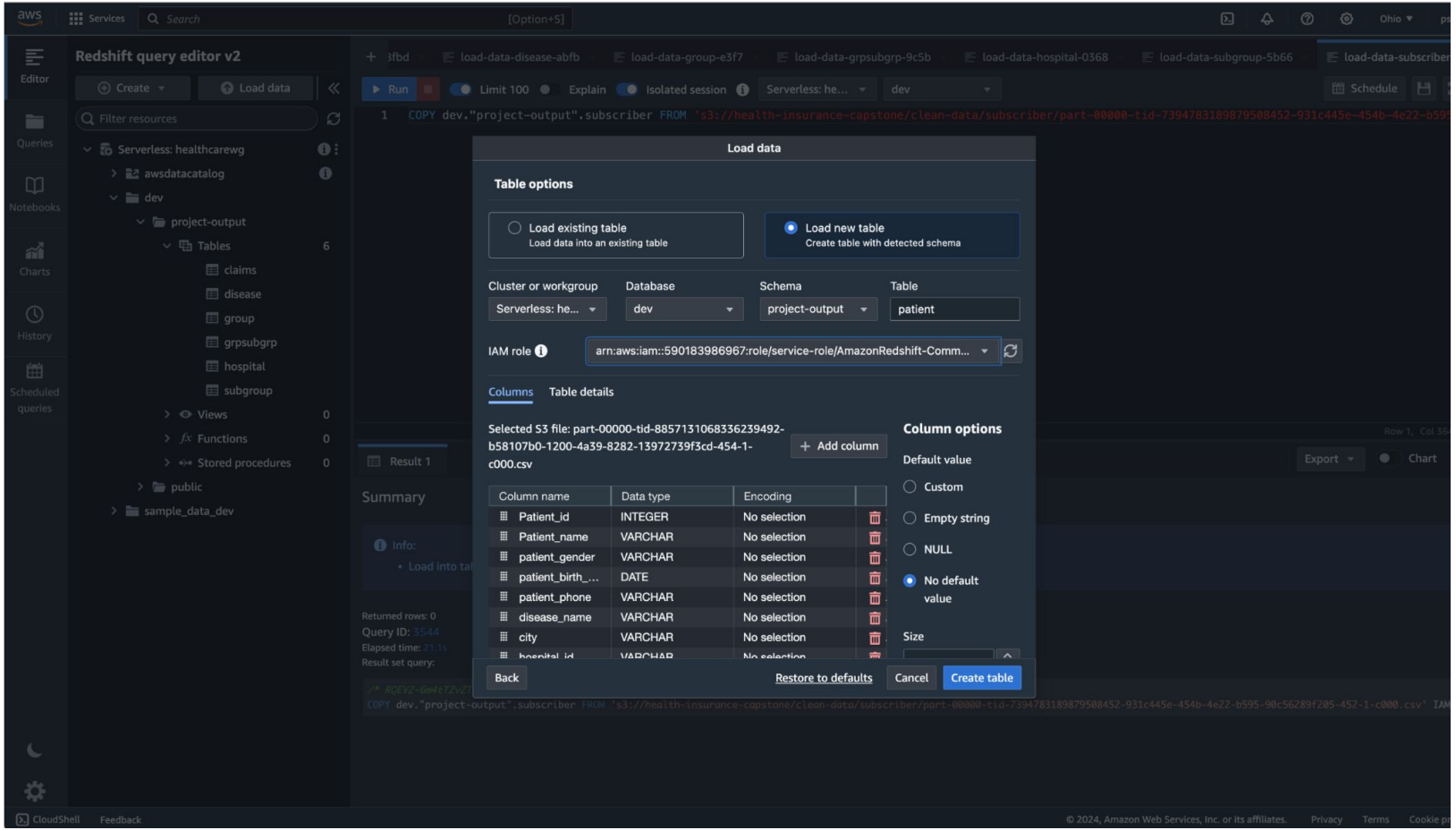
- ▸ Job 259 View (Stages: 1/1)
- ▸ Job 260 View (Stages: 1/1)
- ▸ Job 261 View (Stages: 1/1, 1 skipped)
- ▸ Job 262 View (Stages: 1/1)
- ▸ Job 263 View (Stages: 1/1, 1 skipped)
- ▸ Job 264 View (Stages: 1/1)
- ▸ Job 265 View (Stages: 1/1, 1 skipped)
- ▸ Job 266 View (Stages: 1/1)
- ▸ Job 267 View (Stages: 1/1, 1 skipped)
- ▸ Job 268 View (Stages: 1/1)
- ▸ Job 269 View (Stages: 1/1, 1 skipped)
- ▸ Job 270 View (Stages: 1/1)
- ▸ Job 271 View (Stages: 1/1, 1 skipped)
- ▸ Job 272 View (Stages: 1/1)

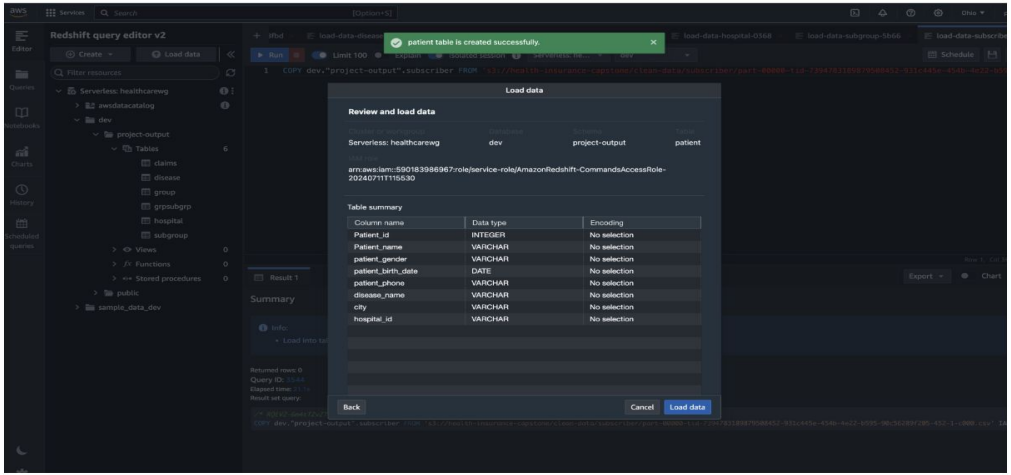

Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3 > Buckets > health-insurance-capstone > clean-data/
clean-data/
Copy S3 URI
Objects
Properties
Objects (8) info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class
claims/
disease/
group/
grpsubgrp/
hospital/
patient/
subgroup/
subscriber/
Folder

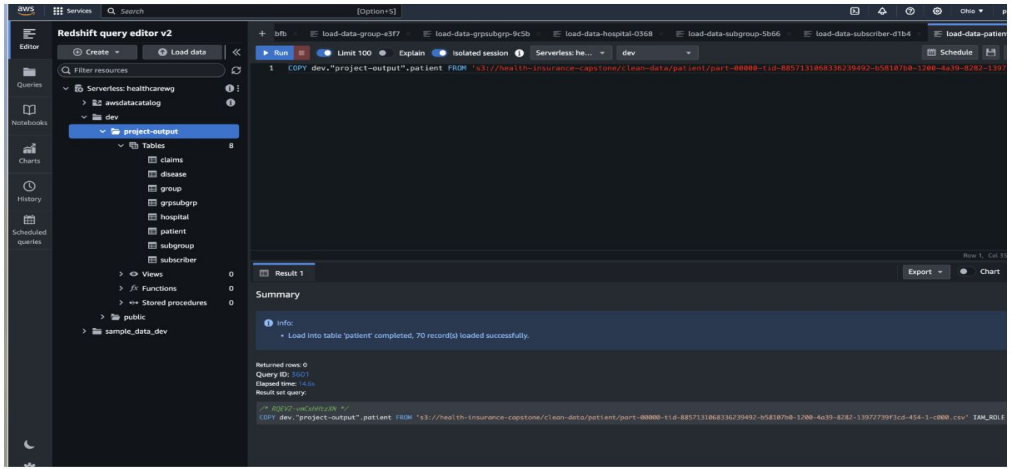

Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3 > Buckets > health-insurance-capstone > clean-data/ > claims/
claims/
Copy S3 URI
Objects
Properties
Objects (6) info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class
_committed_2398982369503752359
_committed_5229221329375621544
_started_2398982369503752359
_started_5229221329375621544
_SUCCESS
part-00000-tid-5229221329375621544-2338baa3-3ed0-4b0f-ac7d-535092a81546-223-1-c000.csv
July 10, 2024, 18:45:40 (UTC-05:00)
July 10, 2024, 18:45:39 (UTC-05:00)
212.0 B
0 B
4.7 KB
Standard
csv

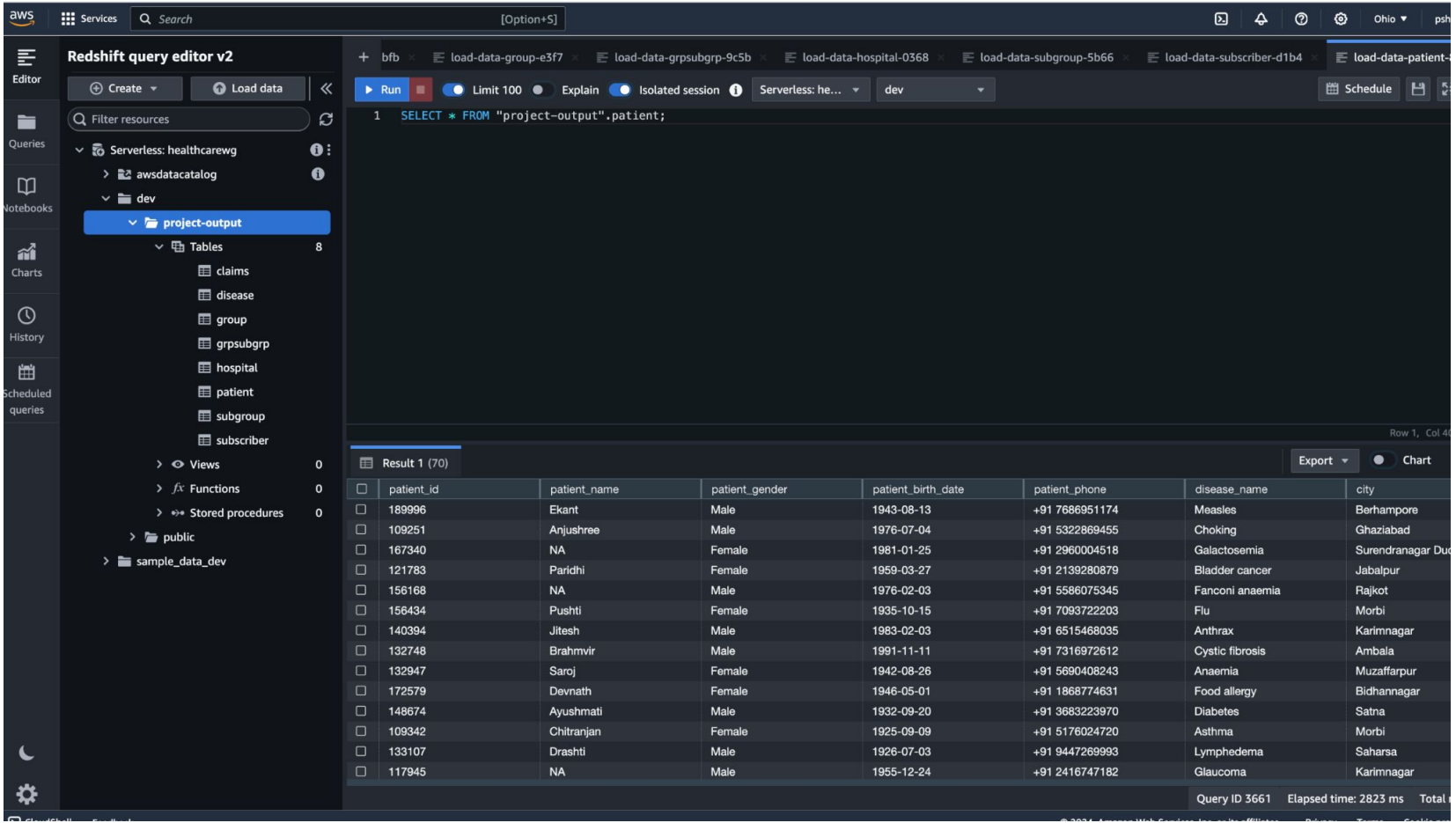
Upload cleaned data corresponding to each data set into a redshift table




Redshift query editor v2
Load data
Table options
Load existing table
Load data into an existing table
Load new table
Create table with detected schema
Cluster or workgroup
Database
Schema
Table
Serverless: he...
dev
project-output
patient
IAM role
arn:aws:iam::590183986967:role/service-role/AmazonRedshift-Comm...
Columns
Table details
Selected S3 file: part-00000-tid-885713106833623949 2-b58107b0-1200-4a39-8282-13972739f3cd-454-1-c000.csv
Add column
Column options
Default value
Custom
Empty string
NULL
No default value
Size
Column name
Data type
Encoding
Patient_id
INTEGER
No selection
Patient_name
VARCHAR
No selection
patient_gender
VARCHAR
No selection
patient_birth_...
DATE
No selection
patient_phone
VARCHAR
No selection
disease_name
VARCHAR
No selection
city
VARCHAR
No selection
Back
Restore to defaults
Cancel
Create table


Redshift query editor v2
patient table is created successfully.
Load data
Review and load data
Serverless: healthcarewg
dev
project-output
patient
arn:aws:iam::590183986967:role/service-role/AmazonRedshift-CommandsAccessRole-20240711T115530
Table summary
Column name
Data type
Encoding
Patient_id
INTEGER
No selection
Patient_name
VARCHAR
No selection
patient_gender
VARCHAR
No selection
patient_birth_date
DATE
No selection
patient_phone
VARCHAR
No selection
disease_name
VARCHAR
No selection
city
VARCHAR
No selection
hospital_id
VARCHAR
No selection
Back
Cancel
Load data


Redshift query editor v2
Serverless: healthcarewg
awsdatacatalog
dev
project-output
Tables
claims
disease
group
grpsubgrp
hospital
patient
subgroup
subscriber
Views
Functions
Stored procedures
public
sample_data_dev
COPY dev."project-output".patient FROM 's3://health-insurance-capstone/clean-data/patient/part-00000-tid-8857131068336239492-b58107b0-1200-4a39-8282-1397
Result 1
Summary
Info:
Load into table 'patient' completed, 70 record(s) loaded successfully.
Returned rows: 0
Query ID:
Elapsed time:
Result set query:


aws
Services
Search
[Option+S]
Ohio
Redshift query editor v2
Editor
Queries
Notebooks
Charts
History
Scheduled queries
Create
Load data
Filter resources
Serverless: healthcarewg
awsdatacatalog
dev
project-output
Tables 8
claims
disease
group
grpsubgrp
hospital
patient
subgroup
subscriber
Views 0
Functions 0
Stored procedures 0
public
sample_data_dev
bfb
load-data-group-e3f7
load-data-grpsubgrp-9c5b
load-data-hospital-0368
load-data-subgroup-5b66
load-data-subscriber-d1b4
load-data-patient-
Run
Limit 100
Explain
Isolated session
Serverless: he...
dev
Schedule
1 SELECT * FROM "project-output".patient;
Row 1, Col 4
Result 1 (70)
Export
Chart
patient_id | patient_name | patient_gender | patient_birth_date | patient_phone | disease_name | city
189996 | Ekant | Male | 1943-08-13 | +91 7686951174 | Measles | Berhampore
109251 | Anjushree | Male | 1976-07-04 | +91 5322869455 | Choking | Ghaziabad
167340 | NA | Female | 1981-01-25 | +91 2960004518 | Galactosemia | Surendranagar Du
121783 | Paridhi | Female | 1959-03-27 | +91 2139280879 | Bladder cancer | Jabalpur
156168 | NA | Male | 1976-02-03 | +91 5586075345 | Fanconi anaemia | Rajkot
156434 | Pushti | Female | 1935-10-15 | +91 7093722203 | Flu | Morbi
140394 | Jitesh | Male | 1983-02-03 | +91 6515468035 | Anthrax | Karimnagar
132748 | Brahmvir | Male | 1991-11-11 | +91 7316972612 | Cystic fibrosis | Ambala
132947 | Saroj | Female | 1942-08-26 | +91 5690408243 | Anaemia | Muzaffarpur
172579 | Devnath | Female | 1946-05-01 | +91 1868774631 | Food allergy | Bidhannagar
148674 | Ayushmati | Male | 1932-09-20 | +91 3683223970 | Diabetes | Satna
109342 | Chitranjan | Female | 1925-09-09 | +91 5176024720 | Asthma | Morbi
133107 | Drashti | Male | 1926-07-03 | +91 9447269993 | Lymphedema | Saharsa
117945 | NA | Male | 1955-12-24 | +91 2416747182 | Glaucoma | Karimnagar
Query ID 3661
Elapsed time: 2823 ms

```python
# 1. Disease with maximum number of claim

from pyspark.sql.functions import col

# Group by disease_name and count the occurrences, then get the one with the maximum count
max_claims_disease_df = claims_df_cleaned.groupBy("disease_name") \
    .count() \
    .orderBy(col("count").desc()) \
    .limit(1)

# Display the result
max_claims_disease_df.show()

# Saving as a permanent table
permanent_table_name = "max_claims_disease"
max_claims_disease_df.write.mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
max_claims_disease_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/max_claims_disease")
```

(15) Spark Jobs

max_claims_disease_df: pyspark.sql.dataframe.DataFrame = [disease_name: string, count: long]

```
+------------+-----+
|disease_name|count|
+------------+-----+
| Pet allergy|    3|
+------------+-----+
```

```python
# 2. Subscribers under age 30 subscribing to any subgroup
from pyspark.sql.functions import datediff, current_date

subscribers_under_30_df = subscriber_df.join(subgroup_df, subscriber_df["Subgrp_id"] == subgroup_df["SubGrp_id"]) \
    .filter(datediff(current_date(), col("Birth_date")) / 365 < 30) \
    .select(subscriber_df["*"])

subscribers_under_30_df.show()

#Saving as a permanent table
permanent_table_name = "subscribers_under_30_df"
subscribers_under_30_df.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)
# Saving data to AWS S3
subscribers_under_30_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/
subscribers_under_30")
```

▸ (4) Spark Jobs

▸ subscribers_under_30_df: pyspark.sql.dataframe.DataFrame = [sub _id: string, first_name: string ... 12 more fields]

```
+----------+------------+---------+-----------------+----------+------+--------------+-------+-----------+--------+---------+--------+----------+--
--------+
|   sub _id|  first_name|last_name|           Street|Birth_date|Gender|         Phone|Country|       City|Zip Code|Subgrp_id|Elig_ind|  eff_date| t
erm_date|
+----------+------------+---------+-----------------+----------+------+--------------+-------+-----------+--------+---------+--------+----------+--
--------+
|SUBID10017|      Bandhu|     Seth|        Varughese|1996-10-15|  Male|+91 0695289163|  India|  Chinsurah|  136713|     S108|       N|2016-10-15|20
18-06-08|
|SUBID10083|  Bhilangana|   Pandit|Ramachandran Path|1995-01-04|Female|+91 6653069630|  India|   Fatehpur|  359466|     S109|       Y|2015-01-04|20
17-10-05|
|SUBID10093|Chandavarman|    Singh|    Sarkar Circle|1997-05-10|Others|+91 6559031791|  India|Navi Mumbai|   83240|     S110|       N|2017-05-10|20
22-08-27|
+----------+------------+---------+-----------------+----------+------+--------------+-------+-----------+--------+---------+--------+----------+--
--------+
```

```python
# 3. Group with maximum subgroup
# Group by Grp_Id and count the subgroups, then get the one with the maximum count
max_subgroup_group_df = grpsubgrp_df.groupBy("Grp_Id") \
    .count() \
    .orderBy(col("count").desc()) \
    .limit(1)

# Display the result
max_subgroup_group_df.show()

# Saving as a permanent table
permanent_table_name = "max_subgroup_group"
max_subgroup_group_df.write.mode("overwrite").saveAsTable(permanent_table_name)
# Saving data to AWS S3
max_subgroup_group_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/max_subgroup_group")
```

(13) Spark Jobs

▸ max_subgroup_group_df: pyspark.sql.dataframe.DataFrame = [Grp_Id: string, count: long]

```
+------+-----+
|Grp_Id|count|
+------+-----+
|GRP104|    2|
+------+-----+
```

```python
from pyspark.sql.functions import col

# Group by hospital_id and count the patients, then get the hospital with the maximum count
most_patients_hospital_df = patient_records_df.groupBy("hospital_id") \
    .count() \
    .orderBy(col("count").desc()) \
    .limit(1)

# Display the result
most_patients_hospital_df.show()

# Saving as a permanent table
permanent_table_name = "most_patients_hospital"
most_patients_hospital_df.write.mode("overwrite").saveAsTable(permanent_table_name)
# Saving data to AWS S3
most_patients_hospital_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/most_patients_hospital")
```

▸ (13) Spark Jobs

▸ most_patients_hospital_df: pyspark.sql.dataframe.DataFrame = [hospital_id: string, count: long]

```
+-----------+-----+
|hospital_id|count|
+-----------+-----+
|      H1017|    9|
+-----------+-----+
```

```python
# 5. Subgroup subscribed most number of times
from pyspark.sql.functions import col

# Group by Subgrp_id and count the occurrences, then get the one with the maximum count
most_subscribed_subgroup_df = subscriber_df.groupBy("Subgrp_id") \
    .count() \
    .orderBy(col("count").desc()) \
    .limit(1)

# Display the result
most_subscribed_subgroup_df.show()

# Saving as a permanent table
permanent_table_name = "most_subscribed_subgroup"
most_subscribed_subgroup_df.write.mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
most_subscribed_subgroup_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/most_subscribed_subgroup")
```

(13) Spark Jobs

most_subscribed_subgroup_df: pyspark.sql.dataframe.DataFrame = [Subgrp_id: string, count: long]

```
+---------+-----+
|Subgrp_id|count|
+---------+-----+
|     S104|   13|
+---------+-----+
```

```python
# 6. Total number of claims which were rejected
from pyspark.sql import Row

total_rejected_claims = claims_df.filter(col("Claim_Or_Rejected") == "Rejected") \
    .count()
#print("Total number of rejected claims:", total_rejected_claims)
# Convert the count to a DataFrame
total_rejected_claims_df = spark.createDataFrame([Row(total_rejected_claims=total_rejected_claims)])

# Display the result
total_rejected_claims_df.show()

# Saving as a permanent table
permanent_table_name = "total_rejected_claims"
total_rejected_claims_df.write.mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
total_rejected_claims_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/
total_rejected_claims")
```

(15) Spark Jobs

▸ total_rejected_claims_df: pyspark.sql.dataframe.DataFrame = [total_rejected_claims: long]

```
+----------------------+
|total_rejected_claims|
+----------------------+
|                     0|
+----------------------+
```

```python
# 7. Join claims_df with subscriber_df to get city information for claims
from pyspark.sql.functions import count, avg, year, current_date, desc
claims_with_city_df = claims_df.join(subscriber_df, claims_df.SUB_ID == subscriber_df["sub _id"], "left")

# City with most claims
city_most_claims = claims_with_city_df.groupBy("City").count().orderBy(desc("count")).first()
print(f"City with most claims: {city_most_claims['City']}")

# Convert to DataFrame
city_most_claims_df = spark.createDataFrame([city_most_claims])

# Display the result
city_most_claims_df.show()

#Saving as a permanent table
permanent_table_name = "city_most_claims"
city_most_claims_df.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
city_most_claims_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/city_most_claims")
```

▸ (7) Spark Jobs

▸ claims_with_city_df: pyspark.sql.dataframe.DataFrame = [Claim_Or_Rejected: string, SUB_ID: string ... 20 more fields]

▸ city_most_claims_df: pyspark.sql.dataframe.DataFrame = [City: string, count: long]

```
City with most claims: Mysore
+------+-----+
|  City|count|
+------+-----+
|Mysore|    2|
+------+-----+
```

```python
# 8. Government or private policy subscriptions
policy_type_subscription = subscriber_df.groupBy("Elig_ind").count().orderBy(desc("count"))
policy_type_subscription.show()

#Saving as a permanent table
permanent_table_name = "policy_type_subscription"
policy_type_subscription.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
policy_type_subscription.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/policy_type_subscription")
```

(6) Spark Jobs

policy_type_subscription: pyspark.sql.dataframe.DataFrame = [Elig_ind: string, count: long]

```
+---------+-----+
|Elig_ind|count|
+---------+-----+
|       N|   50|
|       Y|   46|
|    null|    4|
+---------+-----+
```

```python
# 9. Average monthly premium paid by subscribers
avg_monthly_premium = subgroup_df.select(avg("Monthly_Premium")).first()
print(f"Average monthly premium paid by subscribers: {avg_monthly_premium['avg(Monthly_Premium)']}")

# Convert to DataFrame
avg_monthly_premium_df = spark.createDataFrame([avg_monthly_premium])

# Display the result
avg_monthly_premium_df.show()

#Saving as a permanent table
permanent_table_name = "avg_monthly_premium"
avg_monthly_premium_df.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
avg_monthly_premium_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/
avg_monthly_premium")
```

▸ (6) Spark Jobs

▸ avg_monthly_premium_df: pyspark.sql.dataframe.DataFrame = [avg(Monthly_Premium): double]

```
Average monthly premium paid by subscribers: 1870.0
+--------------------+
|avg(Monthly_Premium)|
+--------------------+
|              1870.0|
+--------------------+
```

```python
# 10. Most profitable group

most_profitable_group = group_df.groupBy("Grp_Id") \
    .agg(sum("premium_written").alias("total_premium")) \
    .orderBy(col("total_premium").desc()) \
    .first()

print("Most profitable group:", most_profitable_group["Grp_Id"])

# Convert to DataFrame
most_profitable_group_df = spark.createDataFrame([most_profitable_group])

# Display the result
most_profitable_group_df.show()

#Saving as a permanent table
permanent_table_name = "most_profitable_group"
most_profitable_group_df.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
most_profitable_group_df.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/
most_profitable_group")
```

▸ (6) Spark Jobs

▸ most_profitable_group_df: pyspark.sql.dataframe.DataFrame = [Grp_Id: string, total_premium: long]

```
Most profitable group: GRP131
+------+-------------+
|Grp_Id|total_premium|
+------+-------------+
|GRP131|        99000|
+------+-------------+
```

```python
# 11. Patients below age of 18 admitted for cancer
from pyspark.sql.functions import sum as spark_sum

# Filter patients below 18 with cancer
patients_below_18_cancer = patient_records_df.filter(
    (col("disease_name").contains("cancer")) &
    (datediff(current_date(), col("patient_birth_date")) / 365 < 18)
)

# Display the result
patients_below_18_cancer.show()

#Saving as a permanent table
permanent_table_name = "patients_below_18_cancer"
patients_below_18_cancer.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
patients_below_18_cancer.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/
patients_below_18_cancer")
```

▸ (2) Spark Jobs

▸ patients_below_18_cancer: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 6 more fields]

```
+----------+------------+-------------+-----------------+------------+------------+----+-----------+
|Patient_id|Patient_name|patient_gender|patient_birth_date|patient_phone|disease_name|city|hospital_id|
+----------+------------+-------------+-----------------+------------+------------+----+-----------+
+----------+------------+-------------+-----------------+------------+------------+----+-----------+
```

```python
#12 List patients who have cashless insurance and have total charges greater than or equal for Rs. 50,000.
# Joining patients and claims DataFrames
joined_df = patient_records_df_cleaned.join(claims_df_cleaned, patient_records_df_cleaned["Patient_id"] == claims_df_cleaned["patient_id"])
# Applying filters
filtered_df = joined_df.filter(
    (claims_df_cleaned["claim_type"] == "claims of value") &
    (claims_df_cleaned["claim_amount"].cast("int") >= 50000)
)
# Selecting required columns
patients_cashless = filtered_df.select(
    patient_records_df_cleaned["Patient_id"],
    patient_records_df_cleaned["Patient_name"],
    patient_records_df_cleaned["hospital_id"],
    claims_df_cleaned["claim_amount"],
    claims_df_cleaned["claim_type"]
)
# Showing the result
patients_cashless.show()
# Saving as a permanent table
permanent_table_name = "patients_cashless"
patients_cashless.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
patients_cashless.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/patients_cashless")
```

(8) Spark Jobs

- joined_df: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 14 more fields]
- filtered_df: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 14 more fields]
- patients_cashless: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 3 more fields]

```
+----------+------------+-----------+------------+---------------+
|Patient_id|Patient_name|hospital_id|claim_amount|     claim_type|
+----------+------------+-----------+------------+---------------+
|    189996|       Ekant|      H1003|      192381|claims of value|
|    109251|   Anjushree|      H1001|      116937|claims of value|
|    167340|          NA|      H1003|      118628|claims of value|
|    132947|       Saroj|      H1016|      186502|claims of value|
|    172579|     Devnath|      H1019|      168634|claims of value|
|    109342|  Chitranjan|      H1011|      125727|claims of value|
|    146382|  Dharmadaas|      H1019|       75983|claims of value|
```

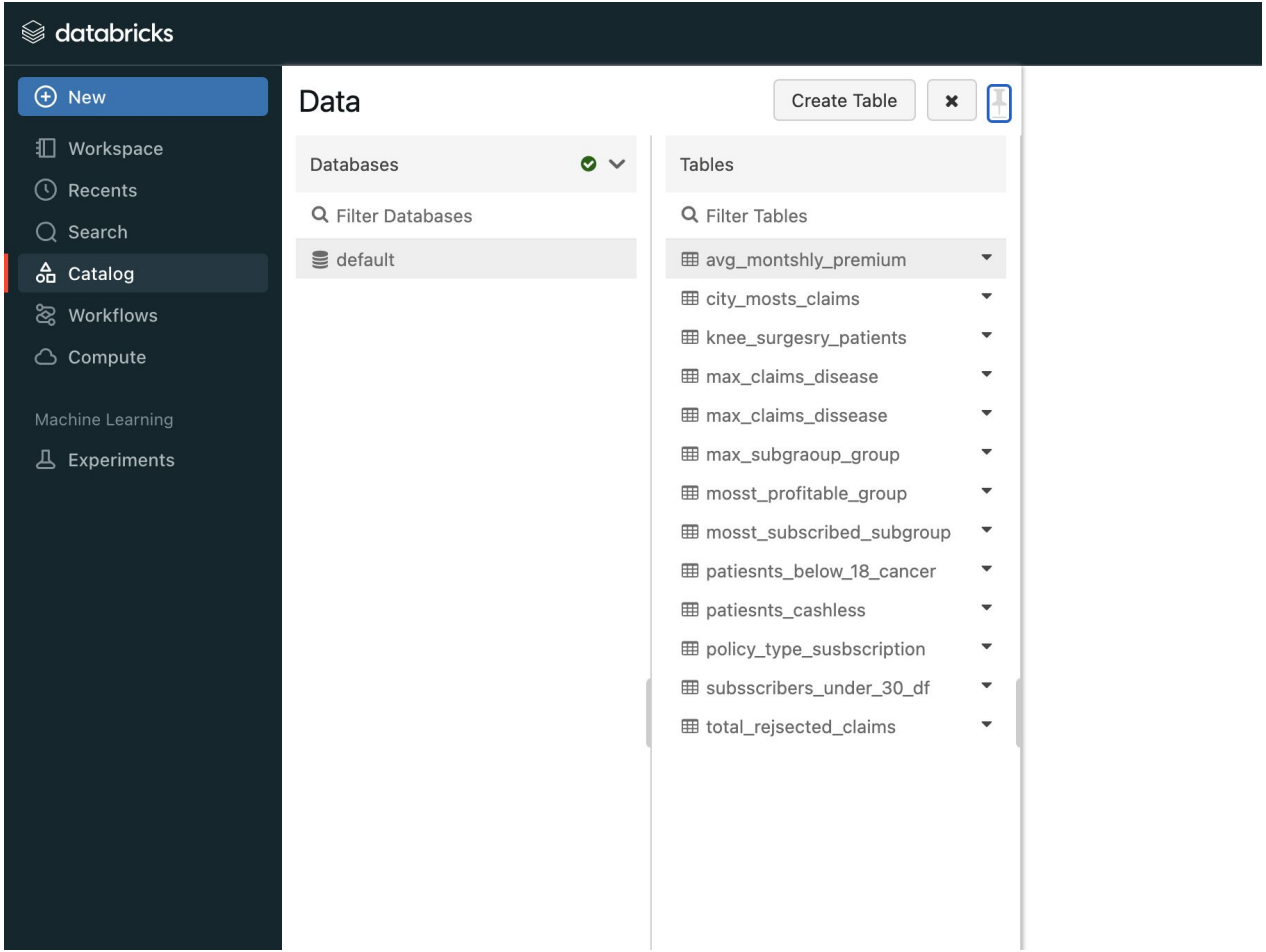
```python
# 13. Female patients over age 40 who underwent knee surgery in the past year
# Filter for female patients over the age of 40
female_patients_over_40 = patient_records_df.filter(
    (col("patient_gender") == "Female") &
    (datediff(current_date(), col("patient_birth_date")) / 365 > 40)
)
#female_patients_over_40.show()
# Filter for knee surgery in the past year
knee_surgery_patients = female_patients_over_40.filter(
    col("disease_name").contains("Knee Surgery"))

# Show the results
knee_surgery_patients.show()

# #Saving as a permanent table
permanent_table_name = "knee_surgery_patients"
knee_surgery_patients.write.format("csv").mode("overwrite").saveAsTable(permanent_table_name)

# Saving data to AWS S3
knee_surgery_patients.write.option("header", "true").mode("overwrite").csv("s3a://health-insurance-capstone/use-case-output/
knee_surgery_patients")
```

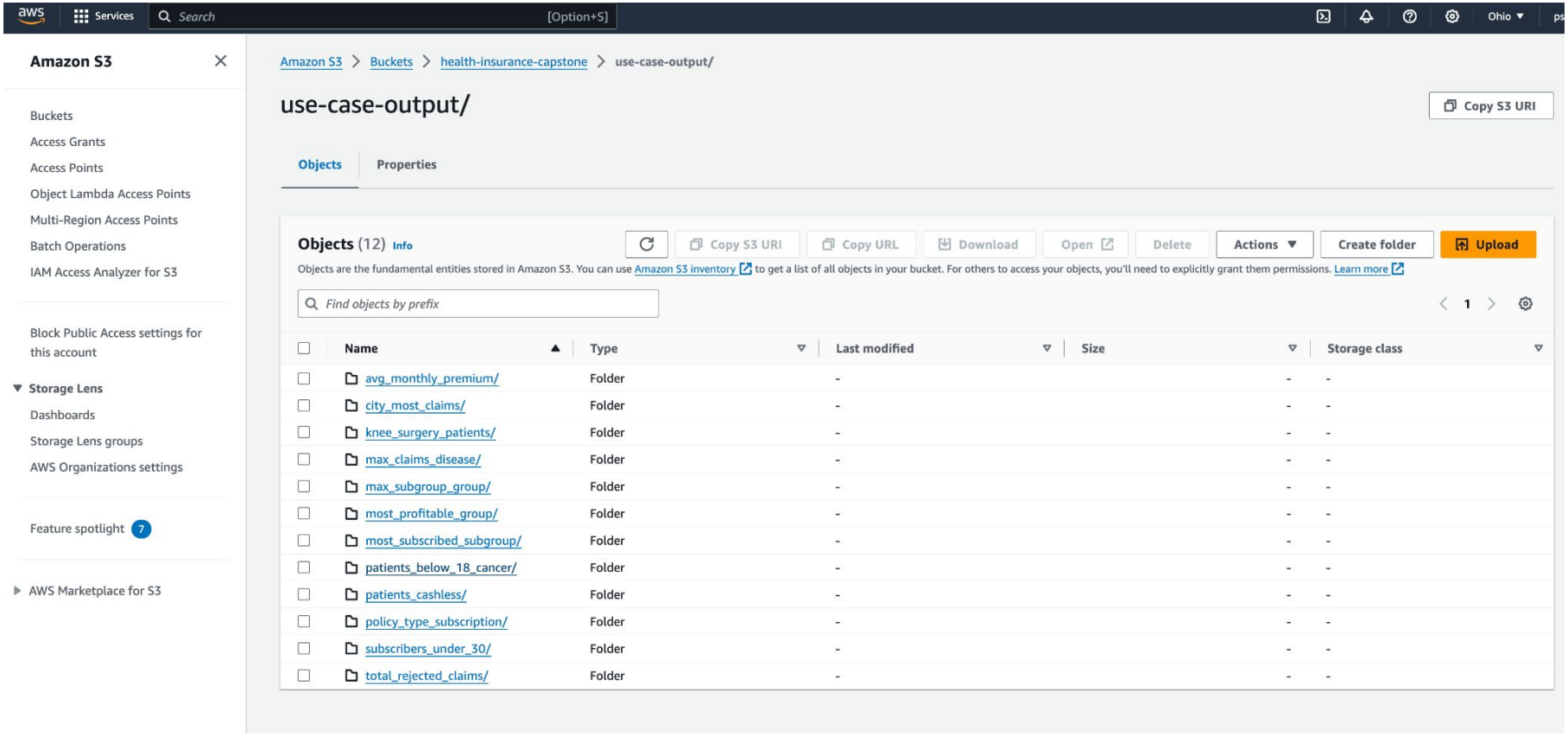
▸ (2) Spark Jobs

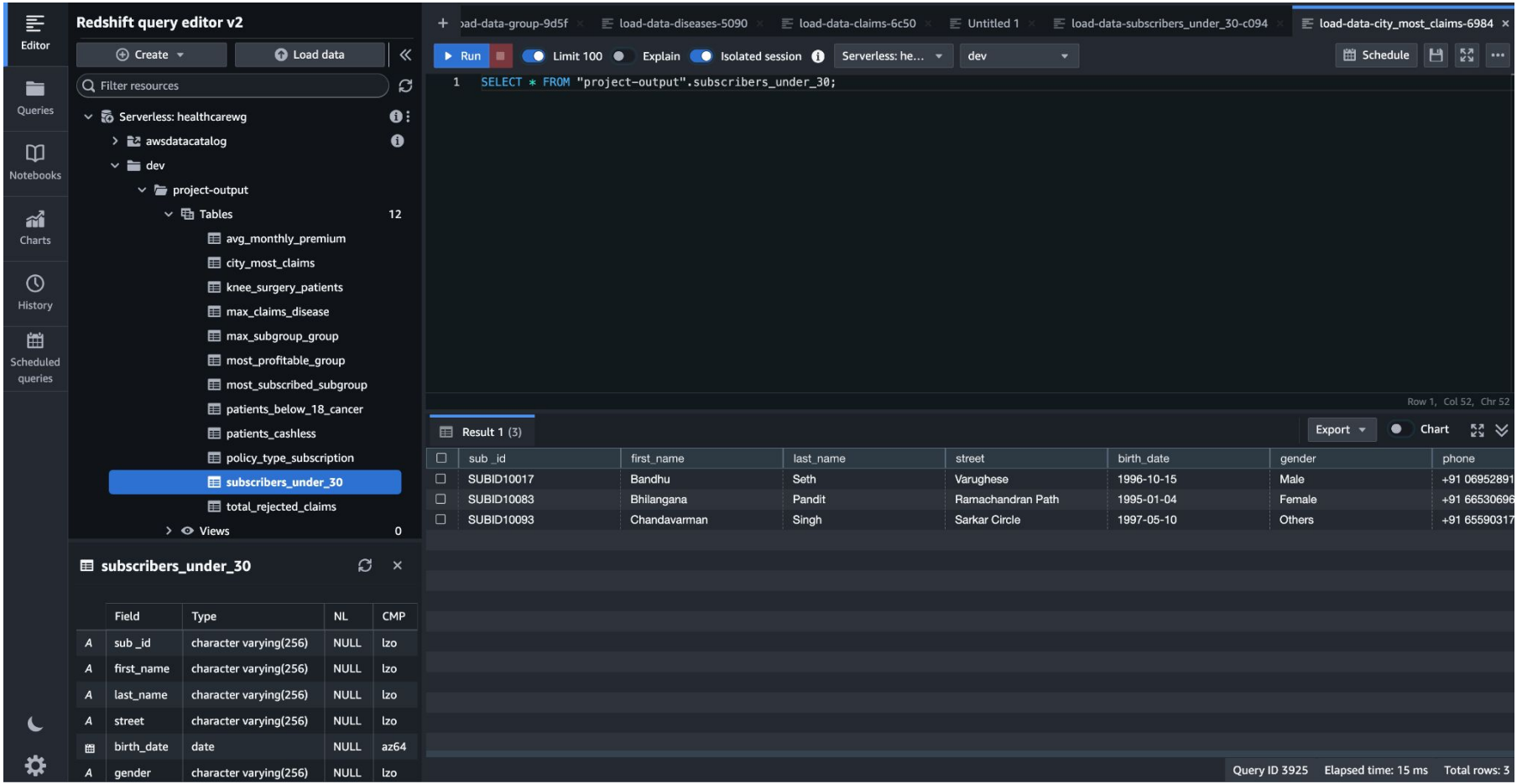
▸ female_patients_over_40: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 6 more fields]

▸ knee_surgery_patients: pyspark.sql.dataframe.DataFrame = [Patient_id: integer, Patient_name: string ... 6 more fields]

```
+----------+------------+--------------+------------------+-------------+------------+----+-----------+
|Patient_id|Patient_name|patient_gender|patient_birth_date|patient_phone|disease_name|city|hospital_id|
+----------+------------+--------------+------------------+-------------+------------+----+-----------+
+----------+------------+--------------+------------------+-------------+------------+----+-----------+
```


databricks
New
Workspace
Recents
Search
Catalog
Workflows
Compute
Machine Learning
Experiments
Data
Create Table
Databases
Filter Databases
default
Tables
Filter Tables
avg_montshly_premium
city_mosts_claims
knee_surgesry_patients
max_claims_disease
max_claims_dissease
max_subgraoup_group
mosst_profitable_group
mosst_subscribed_subgroup
patiesnts_below_18_cancer
patiesnts_cashless
policy_type_susbscription
subsscribers_under_30_df
total_rejsected_claims


aws
Services
Search
[Option+S]
Ohio
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 7
AWS Marketplace for S3
Amazon S3 > Buckets > health-insurance-capstone > use-case-output/
use-case-output/
Copy S3 URI
Objects
Properties
Objects (12) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
avg_monthly_premium/ Folder
city_most_claims/ Folder
knee_surgery_patients/ Folder
max_claims_disease/ Folder
max_subgroup_group/ Folder
most_profitable_group/ Folder
most_subscribed_subgroup/ Folder
patients_below_18_cancer/ Folder
patients_cashless/ Folder
policy_type_subscription/ Folder
subscribers_under_30/ Folder
total_rejected_claims/ Folder


Redshift query editor v2
Editor
Queries
Notebooks
Charts
History
Scheduled queries
Create
Load data
Filter resources
Serverless: healthcarewg
awsdatacatalog
dev
project-output
Tables 12
avg_monthly_premium
city_most_claims
knee_surgery_patients
max_claims_disease
max_subgroup_group
most_profitable_group
most_subscribed_subgroup
patients_below_18_cancer
patients_cashless
policy_type_subscription
subscribers_under_30
total_rejected_claims
Views 0
load-data-group-9d5f
load-data-diseases-5090
load-data-claims-6c50
Untitled 1
load-data-subscribers_under_30-c094
load-data-city_most_claims-6984
Run
Limit 100
Explain
Isolated session
Serverless: he...
dev
Schedule
1 SELECT * FROM "project-output".subscribers_under_30;
Row 1, Col 52, Chr 52
Result 1 (3)
Export
Chart
sub _id | first_name | last_name | street | birth_date | gender | phone
SUBID10017 | Bandhu | Seth | Varughese | 1996-10-15 | Male | +91 06952891
SUBID10083 | Bhilangana | Pandit | Ramachandran Path | 1995-01-04 | Female | +91 66530696
SUBID10093 | Chandavarman | Singh | Sarkar Circle | 1997-05-10 | Others | +91 65590317
subscribers_under_30
Field | Type | NL | CMP
sub _id | character varying(256) | NULL | lzo
first_name | character varying(256) | NULL | lzo
last_name | character varying(256) | NULL | lzo
street | character varying(256) | NULL | lzo
birth_date | date | NULL | az64
gender | character varying(256) | NULL | lzo
Query ID 3925
Elapsed time: 15 ms
Total rows: 3