Health Care Analysis with PySpark

Introduction

Healthcare analysis plays a crucial role in understanding patterns, trends, and disparities within the healthcare system, aiding in decision-making processes for improving patient care, resource allocation, and policy development.

In this project, we focus on leveraging PySpark to perform in-depth analysis on healthcare to gain insights into prescription practices across different cities and states in the United States.

Objectives

The objective of this project is to analyze city and prescriber data to gain insights into prescription practices across different cities. By leveraging PySpark, we aim to preprocess the data, apply various transformations, and load the transformed data into Databricks tables. Additionally, we will build a report in Power BI to visualize the findings.

City Report: We aim to provide insights into healthcare distribution across different cities in the USA. Specifically, we analyze the number of distinct prescribers assigned to each city, the total count of prescriptions (TRX_CNT) prescribed in each city, and the number of ZIP codes within each city. We exclude reporting for cities where no prescribers are assigned, ensuring our analysis focuses on areas actively engaged in healthcare provision.

Prescriber Report: Furthermore, we identify and rank the top 5 prescribers with the highest transaction count (TRX_CNT) within each state. This analysis sheds light on the most prolific prescribers, highlighting potential areas of interest for further investigation or intervention.

Dataset Details

City Dimension File : us_cities_dimension.parquet

Prescriber Fact : USA_Presc_Medicare_Data_2021.csv

Steps Performed

1.Load the dataset containing City details and Prescriber details and check the count details
2.Preprocess the city dimension and prescriber fact data frames
  a. Select only the required columns.
  b. Combine First Name and Last Name and create a new Full Name column in Prescriber fact dataframe
  c. Check for the presence of any Null Values in Prescriber fact dataframe
3.Apply different transformation in city dimension and Presciber dataframes such as:
  a. Create a UDF to calculate the count of zips in each city.
  b. Calculate the number of zips in each city.
  c. Calculate the number of distinct Prescribers assigned and total TRX_CNT prescribed for each city.
  d. Exclude cities with no prescriber assigned from the final city report
  e. Rank the Prescriber dataframe based on highest transaction count according to state
  f. Calculate the Top 5 Prescribers with highest trx_cnt per each state
4.Load the Transformed data into databricks tables
5.Build the report in Power BI

Implementation Details

1. Data Loading

Dataset containing city and prescriber details are uploaded into File store of databricks dbfs:/FileStore/pySparkRealtimeProject

City Dimension Load

2 days ago (18s) 3 Python

```python
##Loading city dimension
city_dim_df =(spark.read
        .format("parquet")
        .load("dbfs:/FileStore/pySparkRealtimeProject/us_cities_dimension.parquet")
        )

city_dim_df.display()
```

(2) Spark Jobs

city_dim_df: pyspark.sql.dataframe.DataFrame = [city: string, city_ascii: string ... 10 more fields]

Table

	city	city_ascii	state_id	state_name	county_fips	county_name	lat
1	New York	New York	NY	New York	36061	New York	40.6943
2	Los Angeles	Los Angeles	CA	California	6037	Los Angeles	34.1139
	Chicago	Chicago	IL	Illinois	17031	Cook	41.8373

Prescriber Fact Load

2 days ago (38s) 5 Python

```python
#USA_Medicare_Prescribers_by_Provider_2021.csv
##Loading prescriber fact
presc_fact_df =(spark.read
        .format("csv")
        .option("header","true")
        .option("inferschema","true")
        .load("dbfs:/FileStore/pySparkRealtimeProject/USA_Medicare_Prescribers_by_Provider_2021.csv")
        )

presc_fact_df.display()
```

(3) Spark Jobs

presc_fact_df: pyspark.sql.dataframe.DataFrame = [PRSCRBR_NPI: integer, Prscrbr_Last_Org_Name: string ... 83 more fields]

Table

	PRSCRBR_NPI	Prscrbr_Last_Org_Name	Prscrbr_First_Name	Prscrbr_MI	Prscrbr_Crdntls	Prscrbr_Gndr	Prscrbr_Ent_Cd	Prscrb
1	1003000126	Enkeshafi	Ardalan	null	M.D.	M	I	6410 R
2	1003000142	Khalil	Rashid	null	M.D.	M	I	4126 N
3	1003000167	Escobar	Julio	E	DDS	M	I	5 Pine
4	1003000175	Reyes-Vasquez	Belinda	null	D.D.S.	F	I	322 N
5	1003000423	Velotta	Jennifer	A	M.D.	F	I	11100
6	1003000480	Rothchild	Kevin	B	MD	M	I	12605
7	1003000522	Weigand	Frederick	J	MD	M	I	1565

3,667 rows | Truncated data | 38.26 seconds runtime Refreshed 2 days ago

2 days ago (3s) 4

```python
#check the count
city_dim_df.count()
```

(2) Spark Jobs

28338

2 days ago (8s)

```python
presc_fact_df.count()
```

(2) Spark Jobs

1287454

2. Data Preprocessing

a. Select only the required columns.

2 days ago (1s) 8 Python

```python
#Data Preprocessing city dimension
    ### Clean df_city DataFrame:
    #1 Select only required Columns
    #2 Convert city, state and county fields to Upper Case

city_dim_pre_df = (city_dim_df.select(
  upper("city").alias("city"),
  "state_id",
  upper("county_name").alias("county_name"),
  "population",
  "zips"
))

city_dim_pre_df.display()
```

(1) Spark Jobs

city_dim_pre_df: pyspark.sql.dataframe.DataFrame = [city: string, state_id: string ... 3 more fields]

Table

	city	state_id	county_name	population	zips
	NEW YORK	NY	NEW YORK	18713220	11229 11226 11225 11224 11222 11221 11220 11385 10169 10168 101

2 days ago (2s) 9 Python

```python
### Clean prescriber_med_fact_df DataFrame:
# 1 Select only required Columns
# 2 Rename the columns
# 3 Add a Country Field 'USA'

presc_fact_pre_df = presc_fact_df.select(
    col("PRSCRBR_NPI").alias("presc_id"),
    col("Prscrbr_Last_Org_Name").alias("presc_lname"),
    col("Prscrbr_First_Name").alias("presc_fname"),
    upper(col("Prscrbr_City")).alias("presc_city"),
    col("Prscrbr_State_Abrvtn").alias("presc_state"),
    col("Prscrbr_Type").alias("presc_spclt"),
    col("Tot_Clms").alias("trx_cnt"),
    col("Tot_Drug_Cst").alias("total_drug_cost"),
    col("Tot_Day_Suply").alias("total_day_supply"))\
    .withColumn("country_name",lit("USA"))


presc_fact_pre_df.display()
```

(1) Spark Jobs

presc_fact_pre_df: pyspark.sql.dataframe.DataFrame = [presc_id: integer, presc_lname: string ... 8 more fields]

Table

	presc_id	presc_lname	presc_fname	presc_city	presc_state	presc_spclt
1	1003000126	Enkeshafi	Ardalan	BETHESDA	MD	Internal Medicine

b. Combine First Name and Last Name and create a new Full Name column in Prescriber fact dataframe

2 days ago (1s) 11 Python

```python
#4.Combine First Name and Last Name
presc_fact_pre_df  = presc_fact_pre_df.\
    withColumn("presc_fullname",concat_ws(" ", "presc_fname", "presc_lname"))

presc_fact_pre_df.display()
```

(1) Spark Jobs

presc_fact_pre_df: pyspark.sql.dataframe.DataFrame = [presc_id: integer, presc_lname: string ... 9 more fields]

Table

		trx_cnt	total_drug_cost	total_day_supply	country_name	presc_fullname
12		21	302.79	341	USA	Juliette Zumwalt
13		297	24801.29	12137	USA	Mike Gallegos

c. Check for the presence of any Null Values in Prescriber fact dataframe

2 days ago (52s) 12

```
# 5. Check and clean all the Null/Nan Values
presc_fact_pre_df.select(
    [
        count(when(isnan(c) | col(c).isNull(), c)).alias(c)
        for c in presc_fact_pre_df.columns
    ]
).display()
```

(2) Spark Jobs

Table

	presc_id	presc_lname	presc_fname	presc_city	presc_state	presc_spclt	trx_cnt	total_drug_cost	total_day_supply	countr
1	0	6	43	0	0	11	0	0	0	0

1 row | 52.30 seconds runtime Refreshed 2 days ago

If any duplicates are found in presc_id drop those records. Delete the records where the PRESC_ID is NULL presc_fact_pre_df = presc_fact_pre_df.dropna(subset="presc_id") Impute TRX_CNT where it is null as avg of trx_cnt for the prescriber eg:presc_fact_pre_df = presc_fact_pre_df.withColumn('trx_cnt', coalesce("trx_cnt",round(avg("trx_cnt").over(spec))))

3. Data Transformation

a. Create a UDF to calculate the count of zips in each city

2 days ago (<1s) 16

```
#Create UDF to calculate the count of zips in each city
@udf(returnType=IntegerType())
def column_split_cnt(column):
    if not column:
        return 0
    return len(column.split(' '))
```

b. Calculate the number of zips in each city.

2 days ago (2s) 17 Python

```
#Calculate the Number of zips in each city
city_dim_sp_df = city_dim_pre_df.withColumn('zip_counts',column_split_cnt(city_dim_pre_df.zips))
city_dim_sp_df.display()
```

(1) Spark Jobs

city_dim_sp_df: pyspark.sql.dataframe.DataFrame = [city: string, state_id: string ... 4 more fields]

Table

	ne	population	zips	zip_counts
1		18713220	11229 11226 11225 11224 11222 11221 11220 11385 10169 10168 10167 10165 10162 10282 10280 10040 10044 11109 11104 11105 11379 11378 11377 11697 11694 11692 11693 11691 10271 10279 10278 10075 10302 10301 10452 11451 10475 10474 10471 10470 10473 10472 11228 11223 10103 11368 11369 11366 11367 11364 11365 11362 11363 11360 11361 10028 10029 10026 10027 10024 10025 10022 10023 10020 10021 11212 11213 11210 11211 11216 11217 11214 11215 11218 11219 10152 10153 10154 10307 10306 10305 11429 10310 10...	310

c. Calculate the number of distinct prescribers assigned and total TRX_CNT prescribed for each city.

2 days ago (23s) 18

```
# Calculate the number of distinct Prescribers assigned for each City and total TRX_CNT prescribed for each city.
presc_fact_grp_df = (presc_fact_pre_df.groupBy(
    "presc_state", "presc_city"
).agg(
    countDistinct("presc_id").alias("presc_counts"), sum("trx_cnt").alias("trx_counts")
))
presc_fact_grp_df.display()
```

(4) Spark Jobs

presc_fact_grp_df: pyspark.sql.dataframe.DataFrame = [presc_state: string, presc_city: string ... 2 more fields]

Table

	presc_state	presc_city	presc_counts	trx_counts
1	CA	ANAHEIM	1255	1617718
2	CA	SALINAS	570	548193

d. Exclude cities with no prescriber assigned from the Final city dataframe

```
#Do not report a city in the final report if no prescriber is assigned to it.
city_dim_join_df = (city_dim_sp_df.join(presc_fact_grp_df,\
    ((city_dim_sp_df.state_id == presc_fact_grp_df.presc_state) & (city_dim_sp_df.city == presc_fact_grp_df.presc_city)),'inner'))
```

city_dim_join_df: pyspark.sql.dataframe.DataFrame = [city: string, state_id: string ... 8 more fields]

e. Rank the Prescriber dataframe by state based on the highest transaction count

```
# Prescriber Report:
#Rank the Prescriber dataframe by state based on the highest transaction count
spec = Window.partitionBy("presc_state").orderBy(col("trx_cnt").desc())
presc_fact_final_df = (presc_fact_pre_df.select("presc_id","presc_fullname","presc_state","country_name","trx_cnt","total_day_supply","total_drug_cost").
withColumn("dense_rank",dense_rank().over(spec))
)

presc_fact_final_df.display()
```

(3) Spark Jobs

presc_fact_final_df: pyspark.sql.dataframe.DataFrame = [presc_id: integer, presc_fullname: string ... 6 more fields]

	presc_id	presc_fullname	presc_state	country_name	trx_cnt	total_day_supply	total_drug_cost	dense_rank
1	1972008811	Daniel Burt	AA	USA	656	26148	120912.93	1
2	1124518162	Mason Tyler	AA	USA	575	22582	20507.14	2
3	1750510327	Nathaniel Rial	AA	USA	560	10922	24546.97	3
4	1033407903	Ryan Mehrer	AA	USA	348	3008	2207.31	4
5	1841794419	Tyler Kendrick	AA	USA	317	15312	22184.53	5
6	1659535557	Charles Tessier	AA	USA	286	9392	16823.19	6
7	1033255906	Todd Sumner	AA	USA	245	1400	953.21	7

10,000 rows | Truncated data | 23.20 seconds runtime   Refreshed 2 days ago

f. Calculate the Top 5 Prescribers with highest trx_cnt per each state

```
#Top 5 Prescribers with highest trx_cnt per each state.

presc_fact_rankfil_df = presc_fact_final_df.filter(col("dense_rank") <= 5)

presc_fact_rankfil_df.display()
```

(2) Spark Jobs

presc_fact_rankfil_df: pyspark.sql.dataframe.DataFrame = [presc_id: integer, presc_fullname: string ... 6 more fields]

	presc_id	presc_fullname	presc_state	country_name	trx_cnt	total_day_supply	total_drug_cost	dense_rank
1	1972008811	Daniel Burt	AA	USA	656	26148	120912.93	1
2	1124518162	Mason Tyler	AA	USA	575	22582	20507.14	2
3	1750510327	Nathaniel Rial	AA	USA	560	10922	24546.97	3
4	1033407903	Ryan Mehrer	AA	USA	348	3008	2207.31	4
5	1841794419	Tyler Kendrick	AA	USA	317	15312	22184.53	5
6	1487964219	Jonelle Ensor	AE	USA	5591	385061	553168.85	1

4. Data Storage

Write the Transformed data into databricks tables.

```
#Writing city_dim_final_df ,presc_fact_rank_df,presc_fact_final_df as tables
city_dim_final_df.write.mode('overwrite').saveAsTable("dim_city_hc")
```

(10) Spark Jobs

```
presc_fact_rankfil_df.write.mode('overwrite').saveAsTable("fact_presc_rank_hc")
```

(11) Spark Jobs

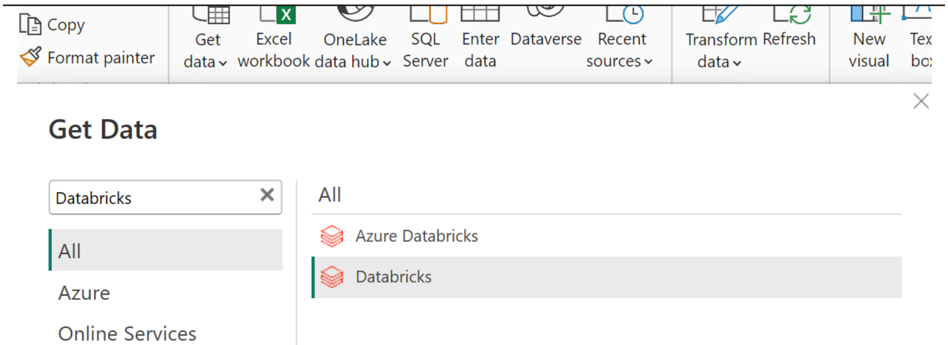
```
presc_fact_final_df.write.mode('overwrite').saveAsTable("fact_presc_hc")
```

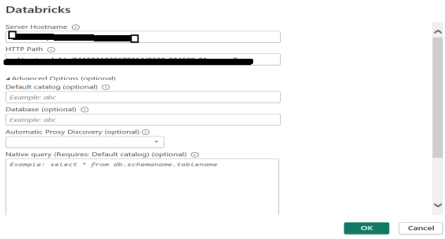
(11) Spark Jobs

5. Reporting

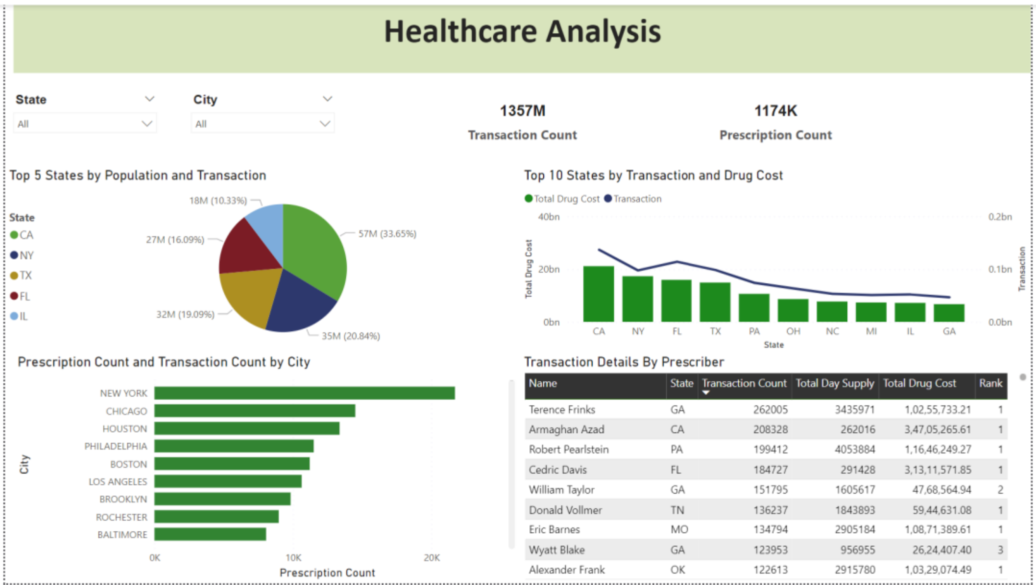
Build the report in Power BI

a. Connect the tables from databricks to Power BI





b. Create a report in Power BI which will showcase the transaction and prescriber details by different metrics



Name	State	Transaction Count	Total Day Supply	Total Drug Cost	Rank
Terence Frinks	GA	262005	3435971	1,02,55,733.21	1
Armaghan Azad	CA	208328	262016	3,47,05,265.61	1
Robert Pearlstein	PA	199412	4053884	1,16,46,249.27	1
Cedric Davis	FL	184727	291428	3,13,11,571.85	1
William Taylor	GA	151795	1605617	47,68,564.94	2
Donald Vollmer	TN	136237	1843893	59,44,631.08	1
Eric Barnes	MO	134794	2905184	1,08,71,389.61	1
Wyatt Blake	GA	123953	956955	26,24,407.40	3
Alexander Frank	OK	122613	2915780	1,03,29,074.49	1

Conclusion

In this PySpark project on healthcare analysis utilizing the City and Prescriber Dataset we have gained a comprehensive understanding of healthcare utilization and prescribing practices.These insights can inform various stakeholders, including policymakers, healthcare administrators, and researchers, in making informed decisions related to resource allocation, policy formulation, and healthcare delivery optimization.