DS 3002 – Capstone Project

Srini Chelimilla

Overview: *This project focuses on a complex ETL process that combines static and dynamic data. I will be using data that models an insurance firm's database. The* ***patients*** *table contains information about the patient, the* ***physicians*** *table contains information about the each physician, the* ***invoices*** *table contains billing information, and the* ***payments*** *JSON files contain information about whether or not each patient has paid their bill. This document contains key steps performed for the project, but the databricks notebook included in the submission and other important files contain detailed steps that may not be mentioned below.*

Setting up Azure Portal

Step 1:

- Created a resource group
- Created sql-server resource and connected it to Azure Data Studio
- Created a sql database resource
- Created a databricks resource

Loading data into Azure Data Studio

Step 1:

- Created an excel sheet with data
- Imported the sheet as csv
- Downloaded the sql server import extension in Azure Data Studio
- Imported csv file (patients) into my database

Step 2:

- Created a new query to populate the date dimension (sql file included in submission)
- Joined the patients table with the date dimension table and updated the patients table to include a date_of_visit_key column with values populated from the date dimension table (sql file included in submission)

Connecting databricks to the sql server

Step 1:

- Defined connection properties

```
jdbcHostname = "ds-3002-sql-server.database.windows.net"
jdbcDatabase = "insurance_pat"
jdbcPort = 1433
jdbcUrl = "jdbc:sqlserver://{0}:{1};database={2}".format(jdbcHostname, jdbcPort, jdbcDatabase)

connectionProperties = {
  "user" : "schelimilla1",
  "password" : "p@$$Wor10",
  "driver" : "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}
```

Step 2:

- Created a dataframe

```
sql_query = """
(SELECT * FROM [dbo].[patients]) patient_dim
"""

df_pat = spark.read.jdbc(url=jdbcUrl, table=sql_query, properties=connectionProperties)
display(df_pat)
```

- Displayed the dataframe

	patient_id	date_of_visit	first_name	last_name	zip_code	doctor_id	invoice_id	date_of_visit_key
1	100	2007-01-23	James	Wilson	68001	25001	10025001	20070123
2	101	2020-08-29	Alex	Smith	20588	25010	10120588	20200829
3	102	2013-05-01	Jack	Taylor	99950	25009	10225009	20130501
4	103	2002-04-20	Ben	Lee	24658	25000	10325000	20020420
5	104	2004-12-25	Eliza	Perez	15295	25017	10425017	20041225
6	105	2012-06-02	Jacob	White	15251	25009	10525009	20120602
7	106	2013-03-30	Sam	Harris	20419	25010	10625010	20130330

Showing all 21 rows.

Loading data into MongoDB Atlas

Step 1:

- Created an account with MongoDB Atlas
- Created a cluster
- Set firewall properties to allow connection from databricks
- Created a database

Step 2:

- Created an excel sheet with data

Connecting databricks to MongoDB Atlas

Step 1:

- Copied the connection string and edited the spark section of the databricks cluster configuration

Step 2:

- Created a data frame

```
%scala
import com.mongodb.spark._

val df_phy = spark.read.format("com.mongodb.spark.sql.DefaultSource").option("database", "insurance_phy").option("collection", "physicians").load()

display(df_phy)
```

- Displaying the data frame

	_id	first_name	last_name	network_status	physician_id
1	{"oid": "627c4d73733c464576a4bd23"}	Mark	Jones	IN NETWORK	25000
2	{"oid": "627c4eab733c464576a4bd24"}	Austin	Smith	OUT OF NETWORK	25001
3	{"oid": "627c4f09733c464576a4bd25"}	Chris	Miller	OUT OF NETWORK	25002
4	{"oid": "627c767cd6450a448be5de8f"}	Evan	Davis	IN NETWORK	25003
5	{"oid": "627c7730d6450a448be5de90"}	Gina	Wilson	IN NETWORK	25004
6	{"oid": "627c7787d6450a448be5de91"}	Tory	Jackson	IN NETWORK	25005
7	{"oid": "627c77c3d6450a448be5de93"}	Mina	Thompson	IN NETWORK	25006
8	{"oid": "627c77f5d6450a448be5de94"}	Ralph	Allen	OUT OF NETWORK	25007
9	{"oid": "627c785cd6450a448be5de95"}	Danielle	Harris	OUT OF NETWORK	25008

- Showing all 20 rows.

Create folder to upload json files

Step 1:

- Went to settings and enabled DBFS
- Created a folder called DS3002_data
- Added API json files
- Added streaming json files

Write data from Azure sql server to databricks

Step 1:

- Defined connection info

```
# Azure SQL server connection info
jdbcHostname = "ds-3002-sql-server.database.windows.net"
jdbcPort = 1433
srcDatabase = "insurance_pat"
dstDatabase = "ins_p"
connectionProperties = {
  "user" : "schelimilla1",
  "password" : "p@$$Wor10",
  "driver" : "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}
```

Step 2:

- Get data from sql database

```
# Fetches data frame from Azure SQL database server
def get_sql_dataframe(host_name, port, db_name, conn_props, sql_query):
    jdbcUrl = f"jdbc:sqlserver://{host_name}:{port};database={db_name}"
    dframe = spark.read.jdbc(url=jdbcUrl, table=sql_query, properties=conn_props)
    return dframe
```

Step 3:

- Create database in databricks and write data from sql server to that database

```
# Fetches data frame from Azure SQL database server
def get_sql_dataframe(host_name, port, db_name, conn_props, sql_query):
    jdbcUrl = f"jdbc:sqlserver://{host_name}:{port};database={db_name}"
    dframe = spark.read.jdbc(url=jdbcUrl, table=sql_query, properties=conn_props)
    return dframe
```

- Queried from the table to make sure table was created

```
%sql
-- Check to see if table was created
SELECT * FROM insurance.dim_patients
```

▸ (1) Spark Jobs

Table | Data Profile

	patient_id	date_of_visit	first_name	last_name	zip_code	doctor_id	invoice_id	date_of_visit_key
1	100	2007-01-23	James	Wilson	68001	25001	10025001	20070123
2	101	2020-08-29	Alex	Smith	20588	25010	10120588	20200829
3	102	2013-05-01	Jack	Taylor	99950	25009	10225009	20130501
4	103	2002-04-20	Ben	Lee	24658	25000	10325000	20020420
5	104	2004-12-25	Eliza	Perez	15295	25017	10425017	20041225
6	105	2012-06-02	Jacob	White	15251	25009	10525009	20120602
7	106	2013-03-30	Sam	Harris	20419	25010	10625010	20130330

Showing all 21 rows.

Step 4:

- Did the same process as above and queried the date dimension table

```
%sql
-- Check to see if table was created
SELECT * FROM insurance.dim_date LIMIT 10
```

▸ (1) Spark Jobs

Table | Data Profile

	DateKey	Date	Day	DaySuffix	Weekday	WeekDayName	WeekDayName_Short	WeekDayName_FirstLetter	DOWInMon
1	20000101	2000-01-01	1	st	7	Saturday	SAT	S	1
2	20000102	2000-01-02	2	nd	1	Sunday	SUN	S	2
3	20000103	2000-01-03	3	rd	2	Monday	MON	M	3
4	20000104	2000-01-04	4	th	3	Tuesday	TUE	T	4
5	20000105	2000-01-05	5	th	4	Wednesday	WED	W	5
6	20000106	2000-01-06	6	th	5	Thursday	THU	T	6
7	20000107	2000-01-07	7	th	6	Friday	FRI	F	7

Showing all 10 rows.

Write data from MongoDB Atlas to databricks

Step 1:

- Defined connection info

```
# MongoDB Atlas connection info
atlas_cluster_name = "dscluster"
atlas_dbname = "insurance_phy"
atlas_user_name = "schelimilla"
atlas_password = "M099oodC10ud"

# Data files info
data_dir = 'dbfs:/FileStore/DS3002_data'
```

Step 2:

- Used the dataframe (df_phy) created in a previous step wrote that data into a table in the insurance database created in the section above

```
df_phy.write.mode("overwrite").saveAsTable("insurance.dim_physicians")
```

- Queried from the table to make sure table was created

```
%sql
-- Check to see if table was created
SELECT insurance.dim_physicians.first_name, insurance.dim_physicians.last_name, insurance.dim_physicians.network_status,
insurance.dim_physicians.physician_id
FROM insurance.dim_physicians
```

▸ (1) Spark Jobs

Table | Data Profile

	first_name	last_name	network_status	physician_id
1	Mark	Jones	IN NETWORK	25000
2	Austin	Smith	OUT OF NETWORK	25001
3	Chris	Miller	OUT OF NETWORK	25002
4	Evan	Davis	IN NETWORK	25003
5	Gina	Wilson	IN NETWORK	25004
6	Tory	Jackson	IN NETWORK	25005
7	Mina	Thompson	IN NETWORK	25006

- Showing all 20 rows.

Reading in API data

Step 1:

- Uploaded JSON file from API into the DS3002 folder in DBFS

Step 2:

- Read the JSON file and created a data frame
- ```
  val df_inv = spark.read.option("multiline",true).json("/FileStore/DS3002_data/invoices.json")
  ```
- Write the data in the data frame to a table in the insurance database in databricks
- ```
  df_inv.write.mode("overwrite").saveAsTable("insurance.dim_invoices")
  ```

Step 3:

- Queried from the table to make sure it was created

```sql
%sql
-- Check to see if table was created
SELECT * FROM insurance.dim_invoices
```

▸ (1) Spark Jobs

Table | Data Profile

	doctor_id	invoice_amount	invoice_id	patient_id	service_date_key
1	25001	15600	10025001	100	20070123
2	25010	401	10120510	101	20200829
3	25009	32760	10225009	102	20130501
4	25000	350	10325000	103	20020420
5	25017	29657	10425017	104	20041225
6	25009	98801	10525009	105	20120602
7	25010	2700	10625010	106	20130330

- Showing all 20 rows.

Created the fact table

<u>Step 1:</u>

- Used sql join statements to create a fact order table
- This fact table included the date of visit, day of week, patient name, network status of physician patient saw, physician name, cost of visit

```sql
%sql
-- Creating the fact table
SELECT insurance.dim_patients.date_of_visit, insurance.dim_date.WeekDayName AS day, insurance.dim_patients.first_name AS patient_first_name,
insurance.dim_patients.last_name AS patient_last_name, insurance.dim_physicians.network_status AS physician_network_status,
insurance.dim_physicians.first_name AS physician_first_name, insurance.dim_physicians.last_name AS physician_last_name,
insurance.dim_invoices.invoice_amount AS bill
FROM insurance.dim_date
JOIN insurance.dim_patients
ON insurance.dim_patients.date_of_visit_key = insurance.dim_date.DateKey
JOIN insurance.dim_physicians
ON insurance.dim_physicians.physician_id = insurance.dim_patients.doctor_id
JOIN insurance.dim_invoices
ON insurance.dim_invoices.service_date_key = insurance.dim_patients.date_of_visit_key
```

- Chose only relevant details and changed column names to visually make sense to a customer
- Resulting table after executing the query:

	date_of_visit	day	patient_first_name	patient_last_name	physician_network_status	physician_first_name	physician_last_name	bill
1	2011-03-12	Saturday	Robert	Baker	IN NETWORK	Gary	Lewis	330
2	2007-01-23	Tuesday	James	Wilson	OUT OF NETWORK	Austin	Smith	15600
3	2001-08-04	Saturday	Emily	Lewis	IN NETWORK	Tory	Jackson	760
4	2006-02-28	Tuesday	Noah	Sanchez	IN NETWORK	Scott	Robinson	230
5	2000-04-13	Thursday	Lily	Jones	IN NETWORK	Stacy	Carter	10
6	2007-07-22	Sunday	Lucas	Robinson	OUT OF NETWORK	Ralph	Allen	28700
7	2002-04-20	Saturday	Ben	Lee	IN NETWORK	Mark	Jones	350

- Showing all 20 rows.

Creating fact order table visualizations

Step 1:

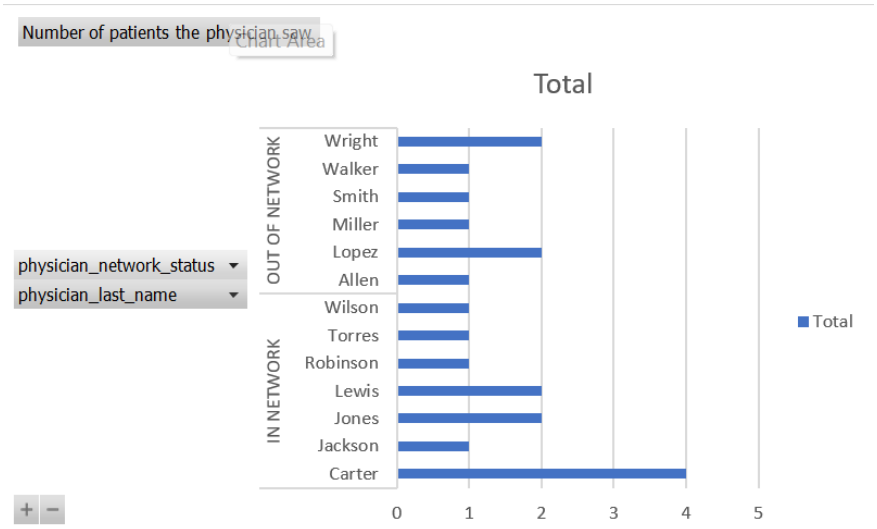
- Downloaded the resulting table from databricks as a csv file
- Imported the file into Excel

Step 2:

- Created a pivot table for the number of patients each physician saw, grouped separately by whether the physician was 'IN NETWORK' or 'OUT OF NETWORK'

Physician	Number of patients the physician saw
IN NETWORK	**12**
Carter	4
Jackson	1
Jones	2
Lewis	2
Robinson	1
Torres	1
Wilson	1
OUT OF NETWORK	**8**
Allen	1
Lopez	2
Miller	1
Smith	1
Walker	1
Wright	2
Grand Total	**20**

- Created a pivot chart for the number of patients each physician saw, grouped separately by whether the physician was 'IN NETWORK' or 'OUT OF NETWORK'

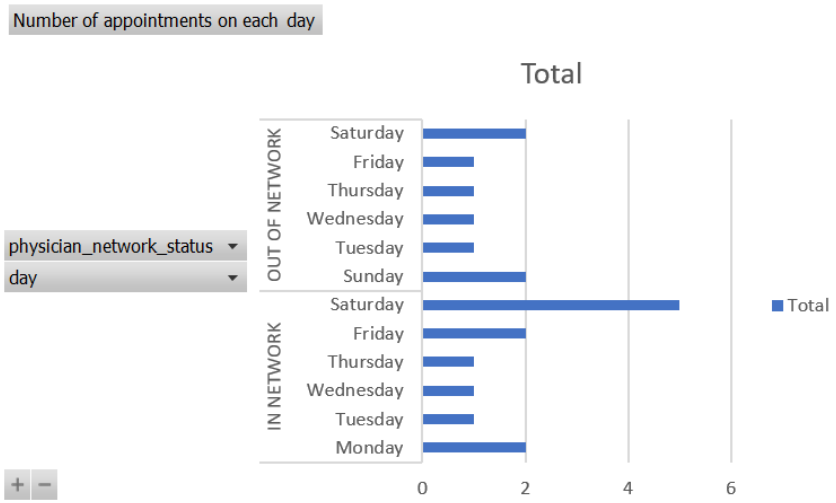
- 

Step 3:

- Created a pivot table for the number of appointments on each day, grouped separately by whether the physician was 'IN NETWORK' or 'OUT OF NETWORK'

Day of Week	Number of appointments on each day
IN NETWORK	**12**
Monday	2
Tuesday	1
Wednesday	1
Thursday	1
Friday	2
Saturday	5
OUT OF NETWORK	**8**
Sunday	2
Tuesday	1
Wednesday	1
Thursday	1
Friday	1
Saturday	2
Grand Total	**20**

- Created a pivot chart for the number of appointments on each day, grouped separately by whether the physician was 'IN NETWORK' or 'OUT OF NETWORK'

- 

Attempt at structured streaming

Step 1:

- Created 4 different JSON files with updated information in each
- Uploaded these files into the DS3002 folder in DBFS

Step 2:

- Created a struct and defined column value types

```
paymentsSchema = StructType([
    StructField("invoice_id", IntegerType(), True),
    StructField("invoice_amount", IntegerType(), True),
    StructField("patient_id", IntegerType(), True),
    StructField("payment_status", StringType(), True)
])
```

- Created a rawDF

```
rawDF = (spark
         .readStream
         .option("maxFilesPerTrigger", 1)
         .schema(paymentsSchema)
         .json(data_dir)
)
```

Step 3:

- Created a bronze folder for streaming checkpoint and output data to store into
- Wrote a streaming query to write data into a delta table

```
# data_dir = 'dbfs:/FileStore/DS3002_data'
outputPathBronze = data_dir + "/payments_data/bronze"

checkpointPathBronze = outputPathBronze + "/checkpoint"
outputPathBronze += "/output.delta"
streamName = "payments_stream"

streamingQuery = (rawDF
                  .writeStream
                  .queryName(streamName)
                  .trigger(processingTime="3 seconds")
                  .format("delta")
                  .option("checkpointLocation", checkpointPathBronze)
                  .outputMode("append")
                  .start(outputPathBronze)
)
```

▸ (1) Spark Jobs

▸ payments_stream (id: 08885e32-ece9-4f8a-8766-433c48ccda36) *Last updated: 2 hours ago*

-
- *I was not able to join the streaming data with my fact orders table*