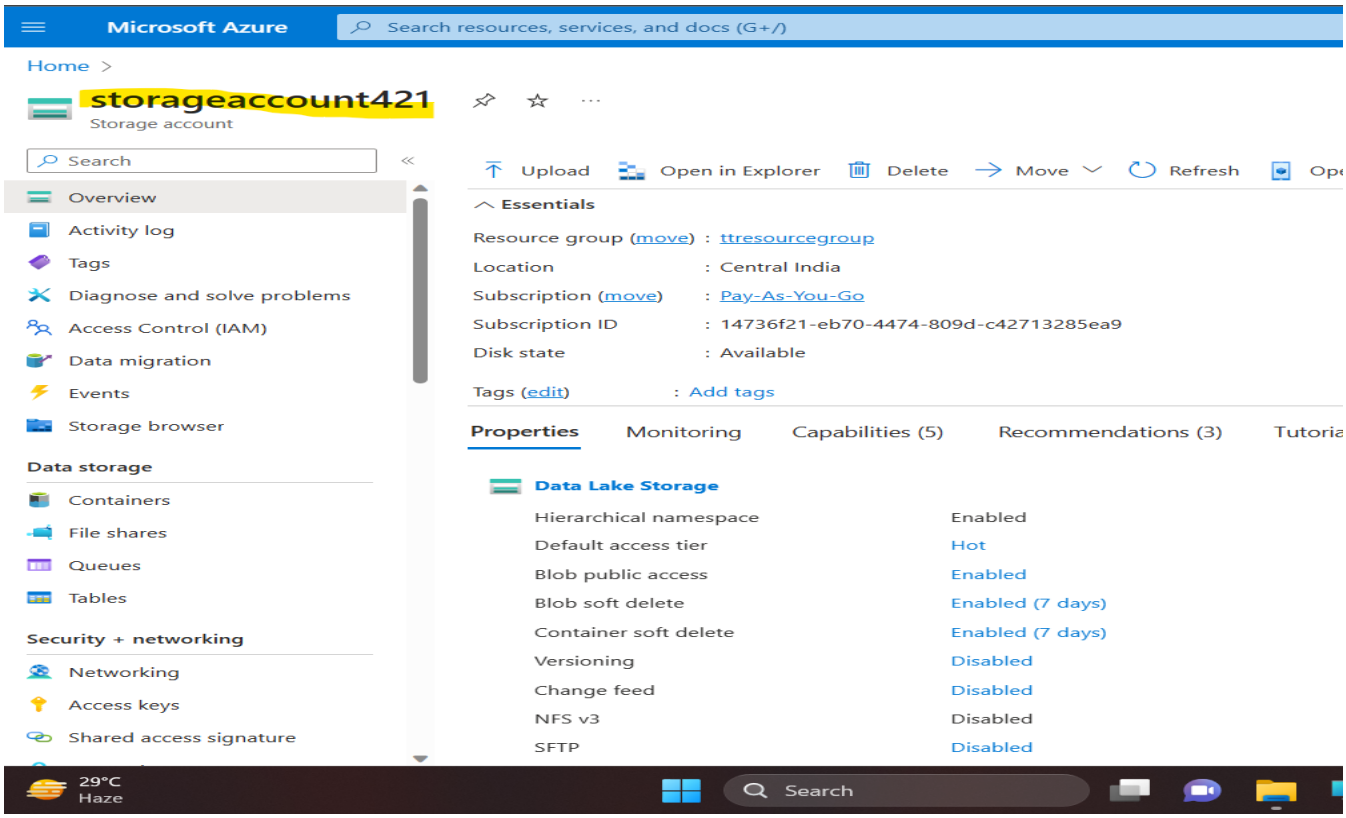
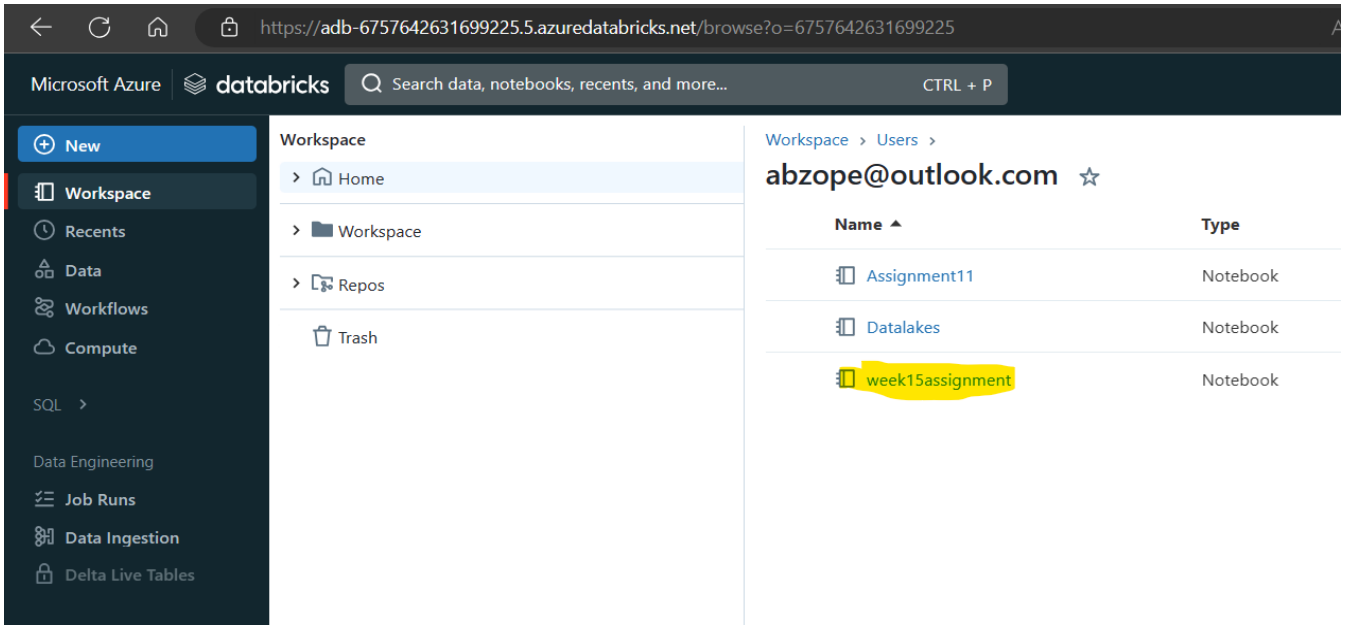
Q1. Perform the following prework

a. Create a storage account in the azure portal
b. Create and launch a databricks workspace in the azure portal
c. Create a single node cluster in the databricks workspace.

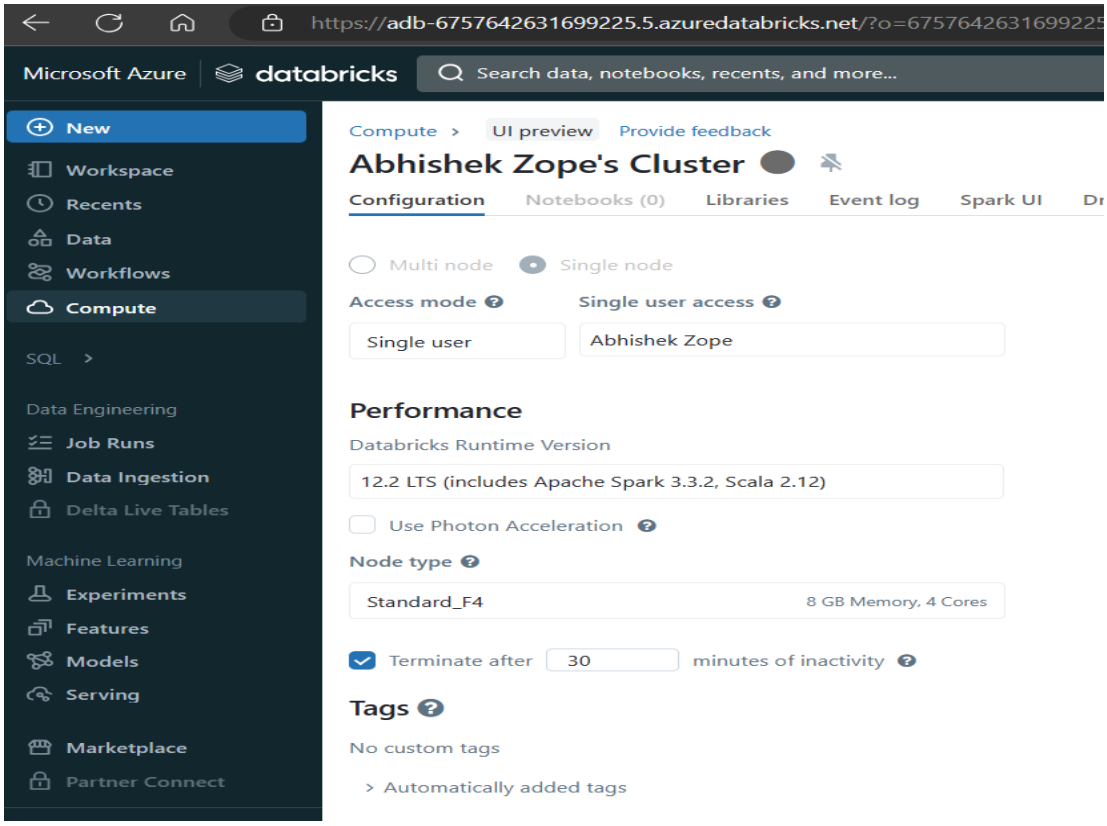
Soln: a.



Soln: b.

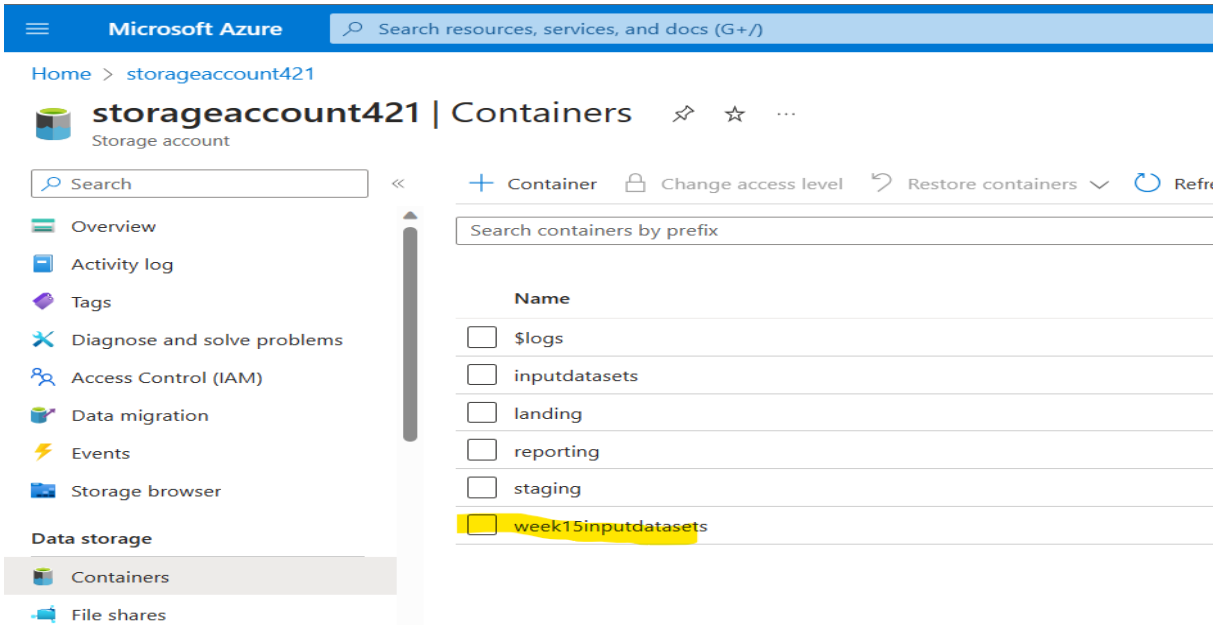


Soln: c.



Q2. Choose any 2 datasets of your choice and upload the datasets into the Storage Account created in the previous step. The files have to uploaded into a container named "week15inputdatasets"

Soln:



Home > storageaccount421 | Containers >

week15inputdatasets
Container

Search

Overview
Diagnose and solve problems
Access Control (IAM)

Settings
Shared access tokens
Manage ACL
Access policy
Properties
Metadata

Upload | Add Directory | Refresh | Rename | Delete | Change tier

Authentication method: Access key (Switch to Azure AD User Account)
Location: week15inputdatasets

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier
_$azuretmpfolder$		
delta		
parquet		
flipkart_products_add.csv	8/14/2023, 7:26:30 PM	Hot (Inferred)
flipkart_products_mod.csv	8/14/2023, 7:22:05 PM	Hot (Inferred)
flipkart_products.csv	8/14/2023, 11:36:58 ...	Hot (Inferred)
flipkart_reviews_mod.csv	8/14/2023, 5:17:20 PM	Hot (Inferred)
flipkart_reviews.csv	8/14/2023, 11:21:11 ...	Hot (Inferred)

Q3.Once the cluster is deployed, create a Notebook and execute the following

a. Create a mount point /mnt/week15assignmentdb to access the files in the container - week15inputdatasets
b. Create Dataframes by reading the data present in Storage account through the mount point created in the previous step.

Soln: a.

week15assignment Python
File Edit View Run Help Last edit was 54 minutes ago Provide feedback
Run all | Terminated | Schedule | Share

Cmd 1

```
dbutils.fs.mount(
    source="wasbs://week15inputdatasets@storageaccount421.blob.core.windows.net",
    mount_point='/mnt/storageaccount421/week15inputdatasets',
    extra_configs= {'fs.azure.account.key.storageaccount421.blob.core.windows.net':'G
                                                                                    '}
    )
```

```
Out[6]: True
```

Command took 10.79 seconds -- by abzope@outlook.com at 14/8/2023, 11:02:03 am on Abhishek Zope's Cluster

Cmd 2

```
dbutils.fs.ls('/mnt/storageaccount421/week15inputdatasets')
```

```
Out[1]: [FileInfo(path='dbfs:/mnt/storageaccount421/week15inputdatasets/delta/', name='delta/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/storageaccount421/week15inputdatasets/flipkart_products.csv', name='flipkart_products.csv', size=1279627, modificatio
nTime=1691993218000),
```

Soln: b

Cmd 4

```
df_products = spark.read.csv('/mnt/storageaccount421/week15inputdatasets/flipkart_products.csv',header = True)
df_reviews = spark.read.csv('/mnt/storageaccount421/week15inputdatasets/flipkart_reviews.csv', header = True)
```

(2) Spark Jobs
df_products: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 6 more fields]
df_reviews: pyspark.sql.dataframe.DataFrame = [id: string, norating1: string ... 7 more fields]

Command took 7.36 seconds -- by abzope@outlook.com at 14/8/2023, 4:57:14 pm on Abhishek Zope's Cluster

Cmd 5

```
display(df_products)
```

(1) Spark Jobs

Table +

	id	title	Rating	gender	pl
1	16695	Fashionable & Comfortable Bellies For Women►►(Brown)	3.9	Women	Fli
2	5120	Combo Pack of 4 Casual Shoes Sneakers For Men►►(Multicolor)	3.8	Men	Fli

Q4. Create a database and create delta tables on the data stored in the Storage account.

a. Create Spark tables in Parquet format
b. Create Spark tables in Delta format
c. How is the Parquet format structuring of files different from that of Delta format? Give a detailed explanation with appropriate examples and diagrams
d. Check on which of the tables, the following query - describe history <table-name> gets executed successfully and why?

Soln: a&b

```
#Q4a.
df_products.write.mode("overwrite").partitionBy("Rating").format("parquet").save("/mnt/storageaccount421/week15inputdatasets/parquet/flipkart_products.parquet"
```

(1) Spark Jobs

Command took 22.06 seconds -- by abzope@outlook.com at 14/8/2023, 11:38:20 am on Abhishek Zope's Cluster

Cmd 8

Python

```
#Q4.b
df_products.write.mode("overwrite").partitionBy("Rating").format("delta").save("/mnt/storageaccount421/week15inputdatasets/delta/flipkart_products.delta")
```

(6) Spark Jobs

Command took 27.98 seconds -- by abzope@outlook.com at 14/8/2023, 11:38:29 am on Abhishek Zope's Cluster

Cmd 9

```
%sql
create database if not exists flipkartdb
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 3.87 seconds -- by abzope@outlook.com at 14/8/2023, 12:04:22 pm on Abhishek Zope's Cluster

```
%sql
create table flipkartdb.fproductstable using parquet location "/mnt/storageaccount421/week15inputdatasets/parquet/flipkart_products.parquet/*"
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 6.54 seconds -- by abzope@outlook.com at 14/8/2023, 12:09:41 pm on Abhishek Zope's Cluster

Cmd 11

```
%sql
select * from flipkartdb.fproductstable
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 5 more fields]

Table +

	id	title	gender	platform	pr
1	2692	Women Pink Flats Sandal	Women	Flipkart	35
2	1668	Men Black, Beige Sandal	Men	Flipkart	80
3	17455	Flip Flops	Women	Flipkart	29
4	17976	Flip Flops	Women	Flipkart	24
5	970	Intraflux M Running Shoes For Men♦♦(Grey)	Men	Flipkart	21

```
%sql
create table flipkartdb.fproductstabledelta using delta location "/mnt/storageaccount421/week15inputdatasets/delta/flipkart_products.delta"
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 1.49 seconds -- by abzope@outlook.com at 14/8/2023, 12:12:50 pm on Abhishek Zope's Cluster

Cmd 13

```
%sql
select * from flipkartdb.fproductstabledelta
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 6 more fields]

Table

	id	title	Rating	gender	pl
1	2692	Women Pink Flats Sandal	4.1	Women	Fli
2	1668	Men Black, Beige Sandal	4.1	Men	Fli
3	17455	Flip Flops	4.1	Women	Fli

Soln: c

Parquet and Delta are both file formats used for storing large-scale data. Parquet is a columnar storage format, which means that it stores data by columns rather than by rows. This makes it efficient for analytical processing, as it allows for faster querying and data retrieval. Delta, on the other hand, is a storage layer built on top of Parquet that provides additional features such as ACID transactions, time travel, and data versioning.

The main difference between the two formats lies in how they structure their files.

Parquet files are composed of row groups, with each row group containing data from the same columns. The same columns are stored together in each row group, which makes it efficient for querying and data retrieval.

Delta files, on the other hand, are built on top of Parquet files but have an additional layer over them that provides advanced features such as a transaction log that keeps track of all changes made to the data. This allows for more flexibility in changing the content of the data, such as updating, deleting, and merging capabilities.

Parquet file is structure:

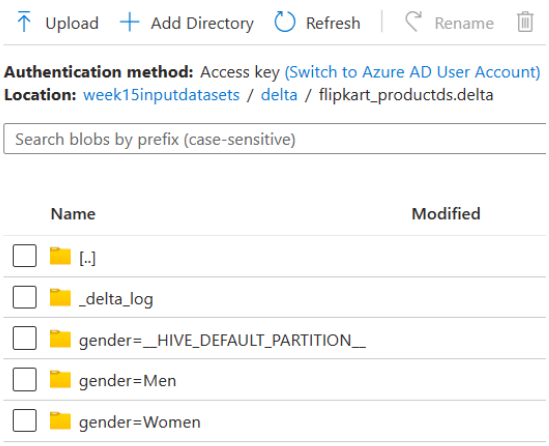
blems

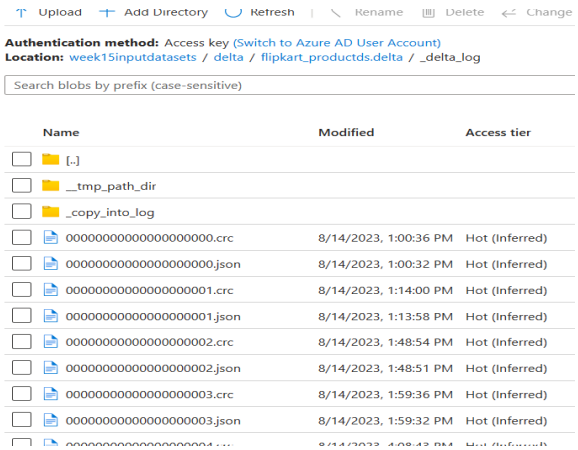
Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archi
[..]			
_committed_5380759779474796050	8/14/2023, 11:38:41 ...	Hot (Inferred)	
_started_5380759779474796050	8/14/2023, 11:38:23 ...	Hot (Inferred)	
_SUCCESS	8/14/2023, 11:38:41 ...	Hot (Inferred)	
part-00000-tid-5380759779474796050-4...	8/14/2023, 11:38:23 ...	Hot (Inferred)	

Delta file structure: I have partitioned it on gender, I had some problems while doing partition on Ratings, so I did on gender.

Delta Table
|
|-- Parquet Files
|-- Transaction Log
  |-- _delta_log
    |-- Version Files
    |-- Checkpoint Files





Soln: d

Delta Lake is an open-source storage layer that brings ACID transactions to Apache Spark and big data workloads. It provides a transaction log that tracks changes to the data, allowing for features such as time travel, audit history, and rollbacks.

The DESCRIBE HISTORY command returns information, including the operation, user, and so on, for each write to a Delta table. This information is retained in the Delta Lake transaction log for a specified period of time.

Other table formats do not have this transaction log and therefore do not have the ability to track changes to the data in the same way. As a result, the DESCRIBE HISTORY command is only

supported for Delta tables.

week15assignment Python

File Edit View Run Help Last edit was 36 minutes ago Provide feedback

Run all | Terminated | Schedule | Share

Cmd 15

```sql
%sql
-- Q4.d
describe history flipkartdb.fproductstable
```

AnalysisException: DESCRIBE HISTORY is only supported for Delta tables.

Command took 1.02 seconds -- by abzope@outlook.com at 14/8/2023, 12:14:55 pm on Abhishek Zope's Cluster

Cmd 16

```sql
%sql
describe history flipkartdb.fproductstabledelta
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table

	version	timestamp	userId	userName	operation	operationParameters	jol
1	0	2023-08-14T06:08:59.000+0000	8606602065517113	abzope@outlook.com	WRITE	{"mode": "Overwrite", "partitionBy": "[\"Rating\"]"}	nu

1 row | 1.77 seconds runtime Refreshed yesterday

Command took 1.77 seconds -- by abzope@outlook.com at 14/8/2023, 12:15:34 pm on Abhishek Zope's Cluster

Cmd 17

Q5. Create a delta table in a single step while writing the data to the dataframe using saveAsTable option

Soln:

Python

```python
#Q.5
df_products.write.mode("overwrite").partitionBy("gender").format("delta").option("path","/mnt/storageaccount421/week15inputdatasets/delta/flipkart_productds.delta").saveAsTable("flipkartdb.fproductsdelta2")
```

(6) Spark Jobs

Command took 9.50 seconds -- by abzope@outlook.com at 14/8/2023, 1:00:27 pm on Abhishek Zope's Cluster

Cmd 18

```sql
%sql
select * from flipkartdb.fproductsdelta2
```

(3) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 6 more fields]

Table

	id	title	Rating	gender	pl
1	16695	Fashionable & Comfortable Bellies For Women**(Brown)	3.9	Women	Fli
2	18391	Cilia Mode Leo Sneakers For Women**(White)	4.4	Women	Fli
3	2692	Women Pink Flats Sandal	4.1	Women	Fli
4	3236	Women Navy Heels Sandal	4	Women	Fli
5	14633	Women Blue Wedges Sandal	3.9	Women	Fli

Type to filter

- hive_metastore
  - default
  - flipkartdb
    - fproductsdelta2
    - fproductstable
    - fproductstabledelta
  - retaildb
  - samples

flipkartdb ›

flipkartdb.fproductsdelta

Comment: Add comment

Columns Sample Data Details

Filter columns...

Q6.

Insert the data into the Delta tables using the following 3 approaches

a.Insert

b.Append

c.Copy

Soln:

```
%sql
-- Q6a.Insert
insert into flipkartdb.fproductsdelta2 values ('12134','Fashion wear','4.8','Women','Flipkart','600','1200','50%')

```

(6) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Table

	num_affected_rows	num_inserted_rows
1	1	1

1 row | 4.83 seconds runtime

Command took 4.83 seconds -- by abzope@outlook.com at 14/8/2023, 1:13:56 pm on Abhishek Zope's Cluster

Cmd 21

```
%sql
select * from flipkartdb.fproductsdelta2 where id = 12134
```

(3) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 6 more fields]

Table

	id	title	Rating	gender	platform	price1	actprice1	Offer
1	12134	Slides	4.1	Women	Flipkart	309	999	69.07%
2	12134	Fashion wear	4.8	Women	Flipkart	600	1200	50%

SQL

```
%sql
describe history flipkartdb.fproductsdelta2
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table

		clusterId	readVersion	isolationLevel	isBlindAppend	operationMetrics	use
1	52"}	0804-054116-qt2qcue7	0	WriteSerializable	true	{"numFiles": "1", "numOutputRows": "1", "numOutputBytes": "2096"}	nul
2	52"}	0804-054116-qt2qcue7	null	WriteSerializable	false	{"numFiles": "3", "numOutputRows": "15730", "numOutputBytes": "328943"}	nul

```python
#Q6b.Append
df_productsadd = spark.read.csv("/mnt/storageaccount421/week15inputdatasets/flipkart_products_add.csv",header = True)
```

(1) Spark Jobs

df_productsadd: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 6 more fields]

Command took 7.93 seconds -- by abzope@outlook.com at 14/8/2023, 1:59:02 pm on Abhishek Zope's Cluster

Cmd 24

```python
df_productsadd.show()
```

(1) Spark Jobs

```
+-----+---------+------+------+--------+------+---------+-----+
|   id|    title|Rating|gender|platform|price1|actprice1|Offer|
+-----+---------+------+------+--------+------+---------+-----+
|11111|Fashion A|   4.1| Women|Flipkart|   500|     1000|  50%|
|22222|Fashion B|   4.2| Women|Flipkart|   500|     1000|  50%|
|33333|Fashion C|   4.3| Women|Flipkart|   500|     1000|  50%|
|44444|Fashion D|   4.4| Women|Flipkart|   500|     1000|  50%|
|55555|Fashion E|   4.6| Women|Flipkart|   500|     1000|  50%|
+-----+---------+------+------+--------+------+---------+-----+
```

Command took 0.52 seconds -- by abzope@outlook.com at 14/8/2023, 1:59:13 pm on Abhishek Zope's Cluster

```python
df_productsadd.write.mode("append").partitionBy("gender").format("delta").save("/mnt/storageaccount421/week15inputdatasets/delta/flipkart_productds.delta")
```

(9) Spark Jobs

Command took 21.02 seconds -- by abzope@outlook.com at 14/8/2023, 1:59:15 pm on Abhishek Zope's Cluster

Cmd 26

SQL

```sql
%sql
describe history flipkartdb.fproductsdelta2
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

	amp	userId	userName	operation	operationParameters
1	8-14T08:29:32.000+0000	8606602065517113	abzope@outlook.com	WRITE	{"mode": "Append", "partitionBy": "[\"gender\"]"}
2	8-14T08:18:51.000+0000	8606602065517113	abzope@outlook.com	COPY INTO	{}
3	8-14T07:43:58.000+0000	8606602065517113	abzope@outlook.com	WRITE	{"mode": "Append", "partitionBy": "[]"}
4	8-14T07:30:32.000+0000	8606602065517113	abzope@outlook.com	CREATE OR REPLACE TABLE AS SELECT	{"isManaged": "false", "description": null, "partitionB

week15assignment Python

File Edit View Run Help Last edit was 21 minutes ago Provide feedback

Run all | Abhishek Zope's Cluster | Schedule | Share

Command took 2.79 seconds -- by abzope@outlook.com at 14/8/2023, 1:59:40 pm on Abhishek Zope's Cluster

Cmd 27

SQL

```sql
%sql
-- Q6c.Copy Into
copy into flipkartdb.fproductsdelta2 from "/mnt/storageaccount421/week15inputdatasets/flipkart_products_add.csv" fileformat = csv
format_options('header' = 'true')
```

(2) Spark Jobs

Job 11 View (Stages: 1/1)

Job 12 View (Stages: 1/1)

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long ... 1 more field]

Table +

Cmd 28

```
%sql
describe history flipkartdb.fproductsdelta2
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

	version	timestamp	userId	userName	operation	operationParameters
1	4	2023-08-14T10:38:40.000+0000	8606602065517113	abzope@outlook.com	COPY INTO	{}
2	3	2023-08-14T08:29:32.000+0000	8606602065517113	abzope@outlook.com	WRITE	{"mode": "Append", "pa

Q7. Depict how schema mis-match is handled in Delta format. Explain by considering a usecase on the data present in the storage account. & Q8. How does Delta format support Schema Evolution? Explain by considering an usecase on the data present in the storage account.

Soln: I had some problem with my existing flipkart dataset, so for this question I have used orders dataset, I copied some new data inside ordersdelta1 table but here i have reduced one column, hence schema mismatch error is triggered, after this have done mergeschema so as an additional column space null is being mapped.
Mistakenly I copied same data twice so we are able to see 2 entries of same.

SQL

```
%sql
-- we have table ordersdelta1 have 4 columns and we want to add additional column, now we want to add that additional column inside table, we pushed csv file with 5 columns and try to copy inside a existing table.
copy into retaildb.ordersdelta1 from "/mnt/storageaccount421/inputdatasets/ordersnew.csv" fileformat = csv format_options('header' = 'true')
-- schema mismatch error
```

(4) Spark Jobs

```
AnalysisException: A schema mismatch detected when writing to the Delta table (Table ID: eb5eda76-d5d2-422c-be04-6acf51a32264).
To enable schema migration using DataFrameWriter or DataStreamWriter, please set:
'.option("mergeSchema", "true")'.
For other operations, set the session configuration
spark.databricks.delta.schema.autoMerge.enabled to "true". See the documentation
specific to the operation for details.
```

Cmd 32

Python

```
# What if we want schema evolution, earlier there were rows with 4 column...now we want to add rows with 5 columns
df2 = spark.read.csv("/mnt/storageaccount421/inputdatasets/ordersnew.csv", header = True)
```

(1) Spark Jobs

df2: pyspark.sql.dataframe.DataFrame = [order_id: string, order_date: string ... 3 more fields]

Cmd 34

```
#evolve schema
df2.write.mode("append").partitionBy("order_status").format("delta").option("mergeSchema","true").save("/mnt/storageaccount421/inputdatasets/delta/orders.delta")
```

(5) Spark Jobs

```
%sql
select * from retaildb.ordersdelta1 where order_id = 3
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [order_id: string, order_date: string ... 3 more fields]

Table +

	order_id	order_date	customer_id	order_status	order_amount
1	3	00:00.0	12111	COMPLETE	21
2	3	00:00.0	12111	COMPLETE	null
3	3	00:00.0	12111	COMPLETE	null

3 rows | 2.84 seconds runtime

Handling of schema mismatch in delta format:

Delta Lake uses schema validation on write, which means that all new writes to a table are checked for compatibility with the target table's schema at write time.
If the schema is not compatible, Delta Lake cancels the transaction altogether (no data is written), and raises an exception to let the user know about the mismatch.
This behavior can be overridden by setting the overwriteSchema option to true when writing data to a Delta table. This will allow you to overwrite the existing schema with the new schema of the data being written.
For example, if you have a Delta table with columns A, B, and C, and you try to write data with columns A, B, and D to this table, you will get a schema mismatch error.
However, if you set the overwriteSchema option to true when writing the data, the existing schema of the Delta table will be overwritten with the new schema of the data being written, and the write operation will succeed.

Schema Evolution:

Delta Lake supports schema evolution, which allows the schema of a table to change over time as new data is added. This is achieved through the use of the mergeSchema option when writing data to a Delta table.
When this option is set to true, any columns that are present in the incoming data but not in the target Delta table are automatically added to its schema. This allows for new columns to be added to the table without having to manually alter the schema.

For example, let's say you have a Delta table with columns A and B, and you want to write data with columns A, B, and C to this table. If you set the mergeSchema option to true when writing the data, the new column C will be automatically added to the Delta table's schema, and the write operation will succeed.

In addition, Delta Lake also provides an autoMerge option that can be set to true to enable schema evolution by default. With this option enabled, you can append data with different schemas without having to set the mergeSchema option every time.

Q9.Depict the internal working of update and delete operation by updating and deleting records of the data present in the storage account.

Soln: Here I have updated gender record to ‘Female’ for this particular record.

```
%sql
--Q9
update flipkartdb.fproductstabledelta set gender = 'Female' where id = 22222
```

(8) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long]

Table

	num_affected_rows
1	1

1 row | 5.96 seconds runtime Refreshed yesterday

Command took 5.96 seconds -- by abzope@outlook.com at 14/8/2023, 7:46:58 pm on Abhishek Zope's Cluster

Cmd 34

```
%sql
select * from flipkartdb.fproductstabledelta where id = 22222
```

(3) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 6 more fields]

Table

	id	title	Rating	gender	platform	price1	actprice1	Offer
1	22222	Fashion B	4.2	Female	Flipkart	500	1000	null

Deleted record.

Cmd 37

```
%sql
delete from flipkartdb.fproductstabledelta where id = 11111
```

(8) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long]

Table

	num_affected_rows
1	2

1 row | 6.02 seconds runtime Refreshed yesterday

Command took 6.02 seconds -- by abzope@outlook.com at 14/8/2023, 7:55:26 pm on Abhishek Zope's Cluster

Describe History

Cmd 38

```
%sql
describe history flipkartdb.fproductstabledelta
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table

	version	timestamp	userId	userName	operation	operationParameters
1	5	2023-08-14T14:25:30.000+0000	8606602065517113	abzope@outlook.com	DELETE	{"predicate": "[\"(cast(id#8542 as int) = 11111)\"]"}
2	4	2023-08-14T14:17:01.000+0000	8606602065517113	abzope@outlook.com	UPDATE	{"predicate": "[\"(cast(id#6794 as int) = 22222)\"]"}

UPDATE and DELETE operations work in a similar way. Delta Lake performs an UPDATE on a table in two steps:

1. Find and select the files containing data that match the predicate, and therefore need to be updated. Delta Lake uses data skipping whenever possible to speed up this process.
2. Read each matching file into memory, update the relevant rows, and write out the result into a new data file.

Once Delta Lake has executed the UPDATE successfully, it adds a commit in the transaction log indicating that the new data file will be used in place of the old one from now on.
The old data file is not deleted, though. Instead, it's simply "tombstoned" — recorded as a data file that applied to an older version of the table, but not the current version. Delta Lake is able to use it to provide data versioning and time travel.

DELETE works just like UPDATE under the hood. Delta Lake makes two scans of the data:
The first scan is to identify any data files that contain rows matching the predicate condition.
The second scan reads the matching data files into memory, at which point Delta Lake deletes the rows in question before writing out the newly clean data to disk.

Q10. Apply NOT NULL and CHECK constraints on the data and demonstrate the behaviour when data violating the constraints are inserted into the delta table.
Soln:

Altering the id column of the flipkartdb.fproductstabledelta table to set it as NOT NULL. This means that the id column will not accept NULL values and any attempt to insert a NULL value into this column will result in an error. Like below

```
%sql
-- Q10
alter table flipkartdb.fproductstabledelta alter column id set not null
```

(6) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 6.18 seconds -- by abzope@outlook.com at 15/8/2023, 4:39:07 pm on Abhishek Zope's Cluster

Cmd 41

```
%sql
insert into flipkartdb.fproductstabledelta(id,title,Rating,gender,platform,price1,actprice1,offer) values(null,'Demo','4.5','Female',
'Flipkart','400','500','50%')
```

(1) Spark Jobs

com.databricks.sql.transaction.tahoe.schema.DeltaInvariantViolationException: NOT NULL constraint violated for column: id.

Command took 1.15 seconds -- by abzope@outlook.com at 15/8/2023, 4:49:40 pm on Abhishek Zope's Cluster

The check constraint is defined as gender in ('Men','Women'). This means that the values in the gender column must be one of the values specified in the list, otherwise an error will occur when attempting to insert or update a row.

Cmd 42

```
%sql
alter table flipkartdb.fproductstabledelta add constraint gender check (gender in ('Men','Women'))
```

(4) Spark Jobs

AnalysisException: 528 rows in spark_catalog.flipkartdb.fproductstabledelta violate the new CHECK constraint (gender in ('Men' , 'Women'))

Command took 3.42 seconds -- by abzope@outlook.com at 15/8/2023, 4:55:45 pm on Abhishek Zope's Cluster

Q11.There have been several changes being made to the table. Say you are required to present the original data without any changes, restore the table to its first version.

Soln:

```
%sql
--Q11
describe history flipkartdb.fproductstabledelta
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [version: long, timestamp: timestamp ... 13 more fields]

Table +

	version	timestamp	userId	userName	operation	operationParameters
7						
8	4	2023-08-14T14:17:01.000+0000	8606602065517113	abzope@outlook.com	UPDATE	{"predicate": "[\"(cast(id#6794 as int) = 22222)\"]
9	3	2023-08-14T14:11:22.000+0000	8606602065517113	abzope@outlook.com	UPDATE	{"predicate": "[\"(cast(id#5047 as int) = 55555)\"]
10	2	2023-08-14T13:57:06.000+0000	8606602065517113	abzope@outlook.com	COPY INTO	{}
11	1	2023-08-14T11:41:55.000+0000	8606602065517113	abzope@outlook.com	COPY INTO	{}
12	0	2023-08-14T06:08:59.000+0000	8606602065517113	abzope@outlook.com	WRITE	{"mode": "Overwrite", "partitionBy": "[\"Rating\"]

12 rows | 1.80 seconds runtime Refreshed 3 hours ago

Command took 1.80 seconds -- by abzope@outlook.com at 15/8/2023, 5:24:36 pm on Abhishek Zope's Cluster

SQL

```
%sql
select * from flipkartdb.fproductstabledelta timestamp as of '2023-08-14T06:08:59.000+0000'
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: string, title: string ... 6 more fields]

Table +

	id	title	Rating	gender	pl
1	2692	Women Pink Flats Sandal	4.1	Women	Fli
2	1668	Men Black, Beige Sandal	4.1	Men	Fli
3	17455	Flip Flops	4.1	Women	Fli
4	17976	Flip Flops	4.1	Women	Fli
5	970	Intraflux M Running Shoes For Men►►(Grey)	4.1	Men	Fli
6	13094	Women Black Flats Sandal	4.1	Women	Fli
7	10916	Jutis For Women►►(Multicolor)	4.1	Women	Fli

10,000 rows | Truncated data | 2.05 seconds runtime

Refreshed 3 hours ago

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

Command took 2.05 seconds -- by abzope@outlook.com at 15/8/2023, 5:26:20 pm on Abhishek Zope's Cluster