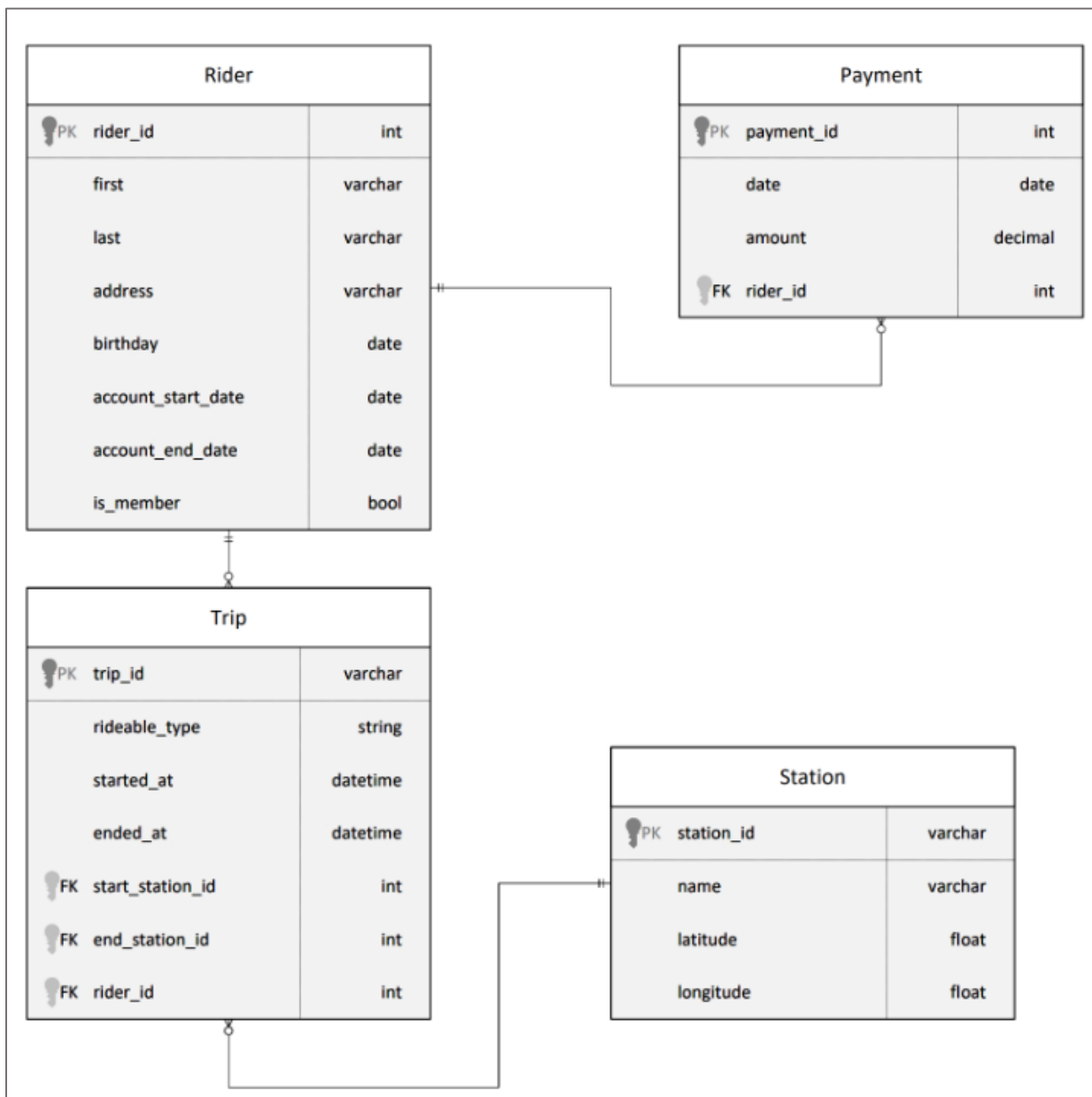


Building an Azure Data Lake for Bike Share Data Analytics

Divvy is a bike sharing program in Chicago, Illinois USA that allows riders to purchase a pass at a kiosk or use a mobile application to unlock a bike at stations around the city and use the bike for a specified amount of time. The bikes can be returned to the same station or to another station. The City of Chicago makes the anonymized bike trip data publicly available for projects like this where we can analyze the data. The dataset looks like this:



The goal of this project is to develop a data lake solution using Azure Databricks using a lake house architecture.

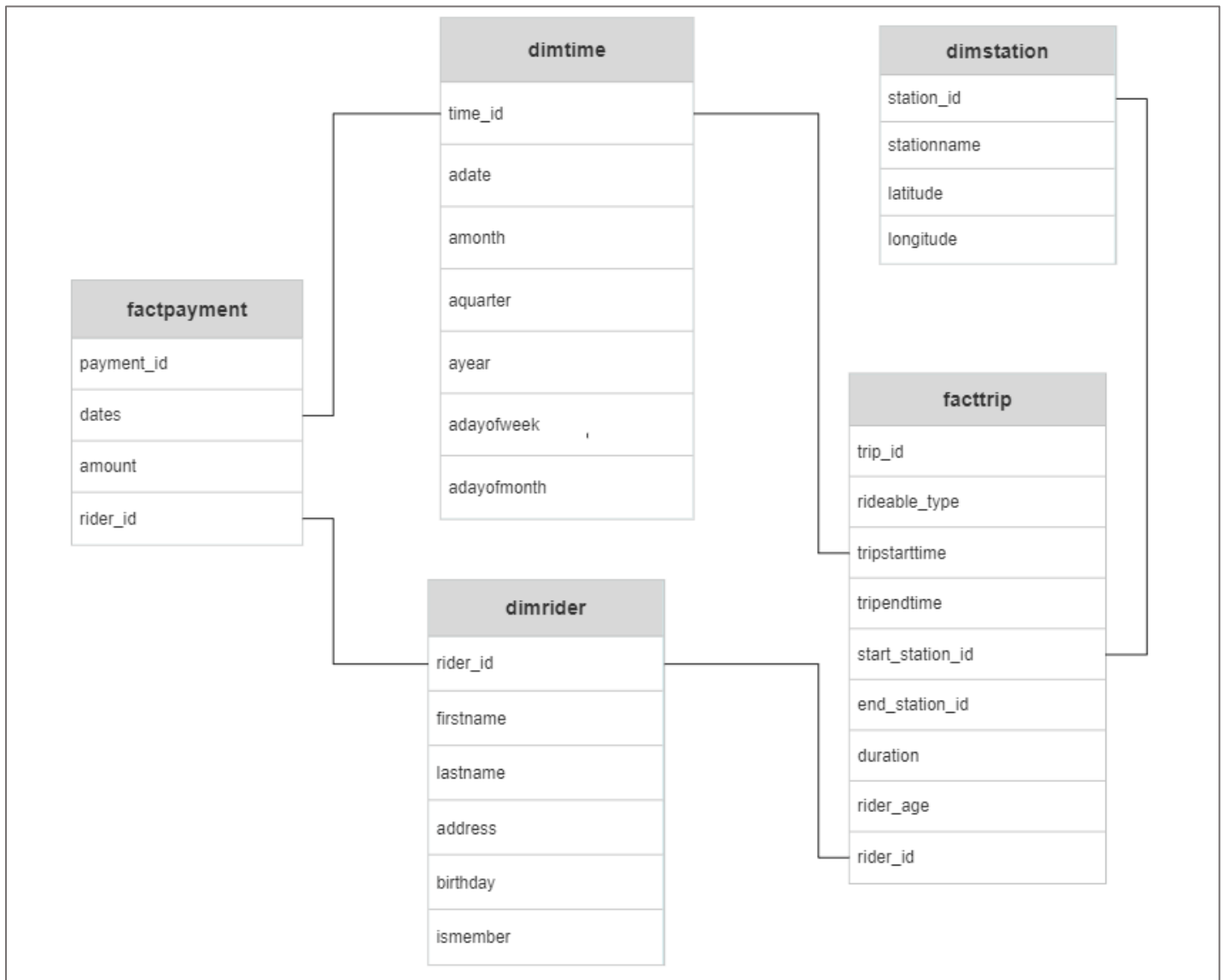
- Design a star schema based on the business outcomes listed below
- Import the data into Azure Databricks using Delta Lake to create a Bronze data store
- Create a gold data store in Delta Lake tables
- Transform the data into the star schema for a Gold data store

The business outcomes we are designing for are as follows:

1. Analyze how much time is spent per ride
 - Based on date and time factors such as day of week and time of day
 - Based on which station is the starting and / or ending station
 - Based on age of the rider at time of the ride
 - Based on whether the rider is a member or a casual rider
2. Analyze how much money is spent
 - Per month, quarter, year
 - Per member, based on the age of the rider at account start
3. Analyze how much money is spent per member
 - Based on how many rides the rider averages per month
 - Based on how many minutes the rider spends on a bike per month

Task 1: Design a star schema

Designing a Star Schema based on the relation diagram and the business problems outlined above:



Star Schema design comprising 2 fact tables, and 3 dimension tables:

FACT TABLES:

- facttrip
- factpayment

DIMENSION TABLES:

- dimrider
- dimtime
- dimstation

Task 2: Create Resources

1. Azure Databricks Workspace

Microsoft Azure

Search resources, services, and docs (G+/)

[Home](#) > [Create a resource](#) > [Azure Databricks](#) >

Create an Azure Databricks workspace ...

Basics

Networking

Advanced

Tags

Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Vocareum-UDA-1

Resource group * ⓘ

Regroup_3uPAJhaz_d

[Create new](#)

Instance Details

Workspace name *

udacitypractice01

✓

Region *

Australia Southeast

▼

Pricing Tier * ⓘ

Standard (Apache Spark, Secure with Azure AD)

▼

Review + create

< Previous


Next : Networking >

Microsoft Azure

Search resources, services, and docs (G+/)

🔍 📄 🔔 ⚙️

[Home](#) >

 **udacitypractice01** ⚙️ ☆ ...

Azure Databricks Service

Search

Delete

Overview

Activity log

Access control (IAM)

Tags

Settings

Virtual Network Peering

Essentials

Status : Active

Resource group : [Regroup_3uPAJhaz_d](#)

Location : Australia Southeast

Subscription : [Vocareum-UDA-1](#)

Subscription ID : 94ec3a64-dcfe-4219-9e29-88221690c382

Tags [\(edit\)](#) : [Click here to add tags](#)

Managed Resource Group : [databricks-rg-udacitypractice01-dvrs3xpnnhwu2](#)

URL : <https://adb-1054427425679412.12.azuredatabricks.net>


Pricing Tier : standard


2. Create a Spark Cluster in the Databricks Workspace


[Clusters](#) / [New Compute](#) [UI Preview](#) [Provide feedback](#)


student_10f0d9bbl8xniydm's Cluster

☒ Multi node ☐ Single node


Access mode 


Single user access 


Single user 


student_10f0d9bbl8xniydm 

Performance

Databricks runtime version 


Runtime: 10.4 LTS (Scala 2.12, Spark 3.2.1) 

☐ Use Photon Acceleration 

Worker type 


Min workers

Max workers


Standard_DS3_v2 14 GB Memory, 4 Cores 


1

2

☐ Spot instances 


Driver type

Same as worker 14 GB Memory, 4 Cores 

☒ Enable autoscaling 

☒ Terminate after

10

 minutes of inactivity 

Tags

Add tags

Key

Value

Add

> Automatically added tags

► Advanced options

Create Cluster

Cancel


Summary

1-2 Workers 14-28 GB Memory 4-8 Cores



1 Driver 14 GB Memory, 4 Cores

Runtime 10.4.x-scala2.12

Standard_DS3_v2 1.5-2.25 DBU/h

Microsoft Azure  Search


CTRL + P


udacitypractice01  student_10f0d9bbl8xniydm_00826... 


Get started

This is your home for all data science and engineering work.


We'll show you how to set up clusters, data and users.


Create a cluster 

Import data 

Invite your team 

Next steps

Ingest data into Delta Lake 





Read documentation 

Compute [UI Preview](#) [Provide feedback](#)

All-purpose clusters Job clusters Pools

Create Cluster

All Created by me Accessible by me

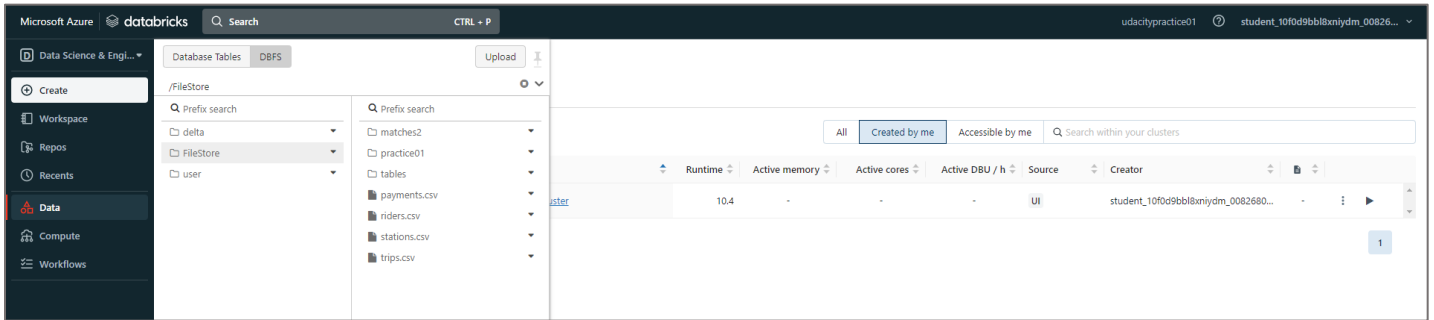
	Name	Runtime	Active memory	Active cores	Active DBU / h	Source	Creator		
	 student_10f0d9bbl8xniydm's Cluster	10.4	28 GB	8 cores	1.5	UI	student_10f0d9bbl8xniydm_0082680...		

1

Task 3: Extract step

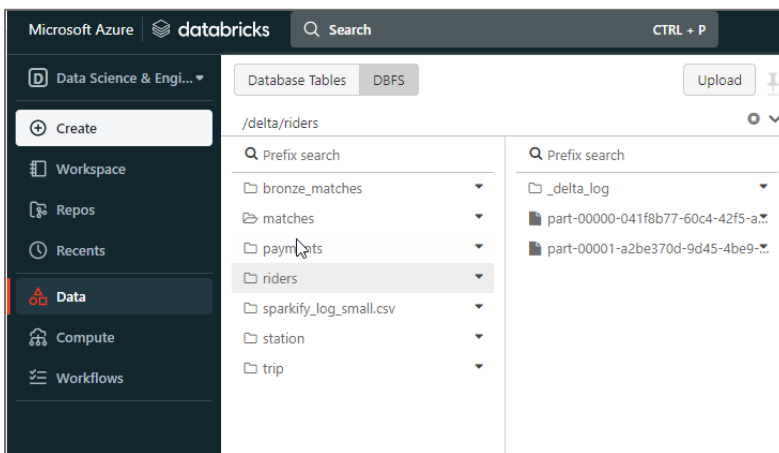
1. Uploaded the 4 csv files to the DBFS Filestore manually

payments
riders
stations
trips

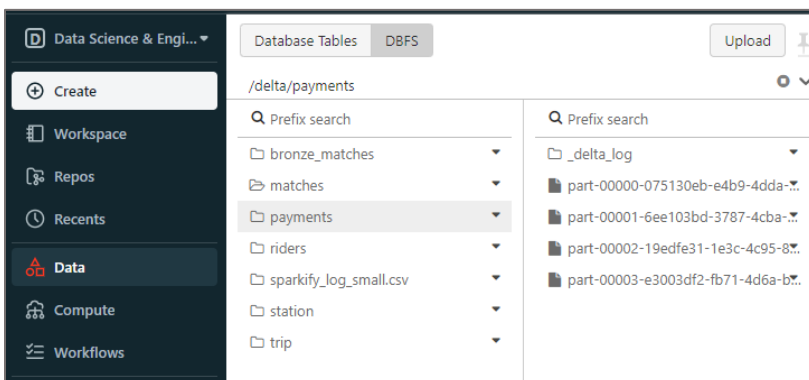


2. Using the Extract Python Notebook, Extracted the data to the DBFS DELTA Storage

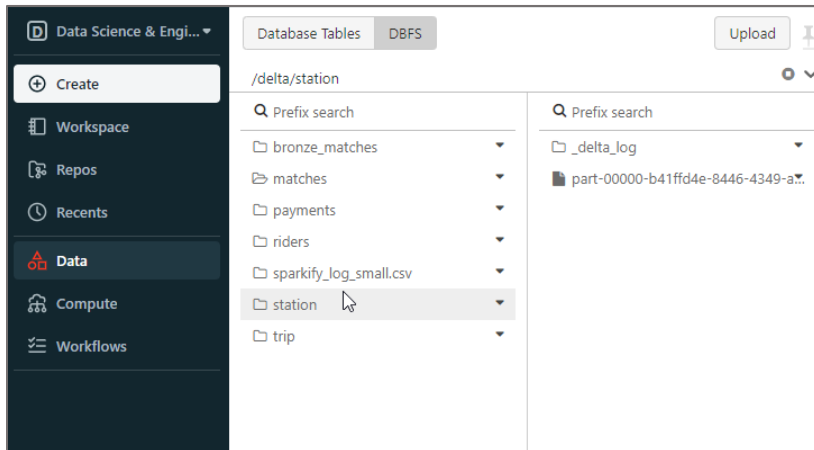
Riders



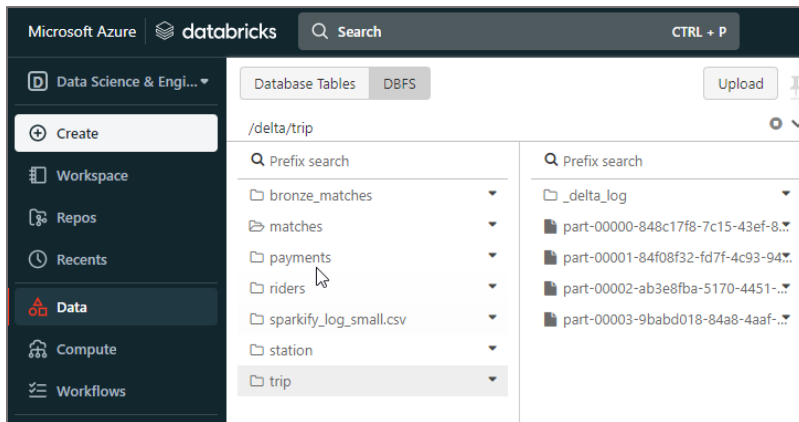
Payments



Station



Trip



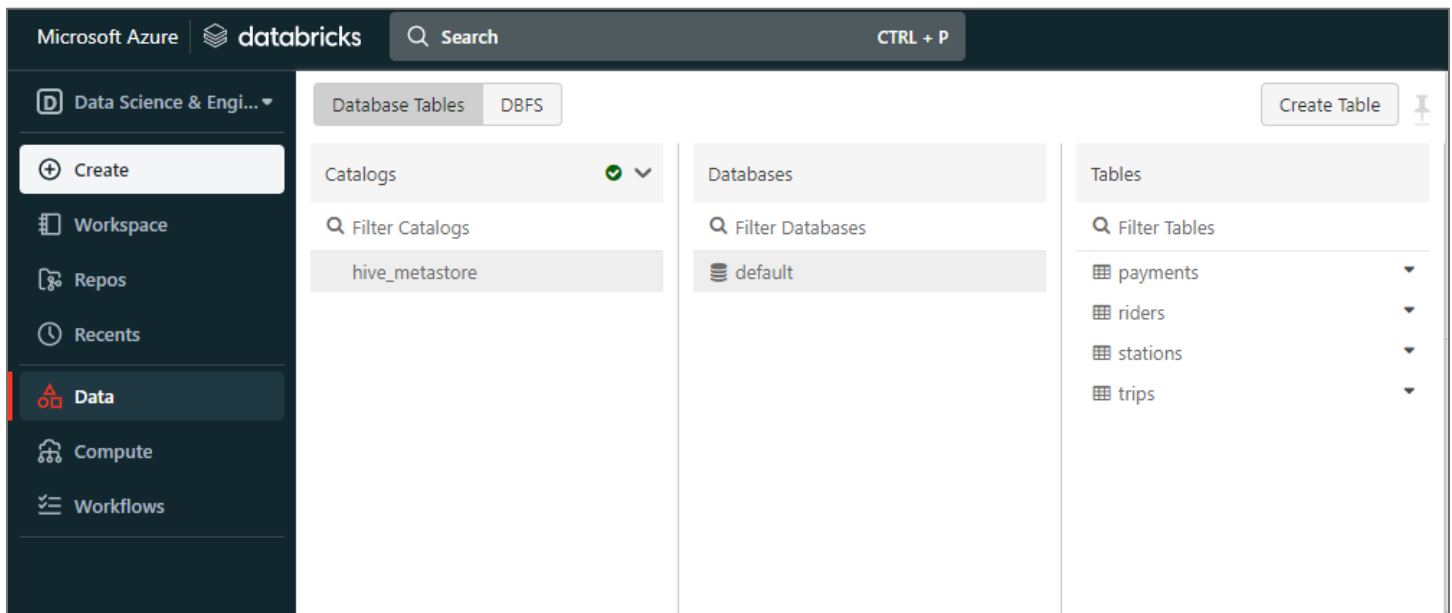
Task 4: Load

-Complete Extraction by loading data from DBFS Delta to DBFS Databasetables

cmd 6

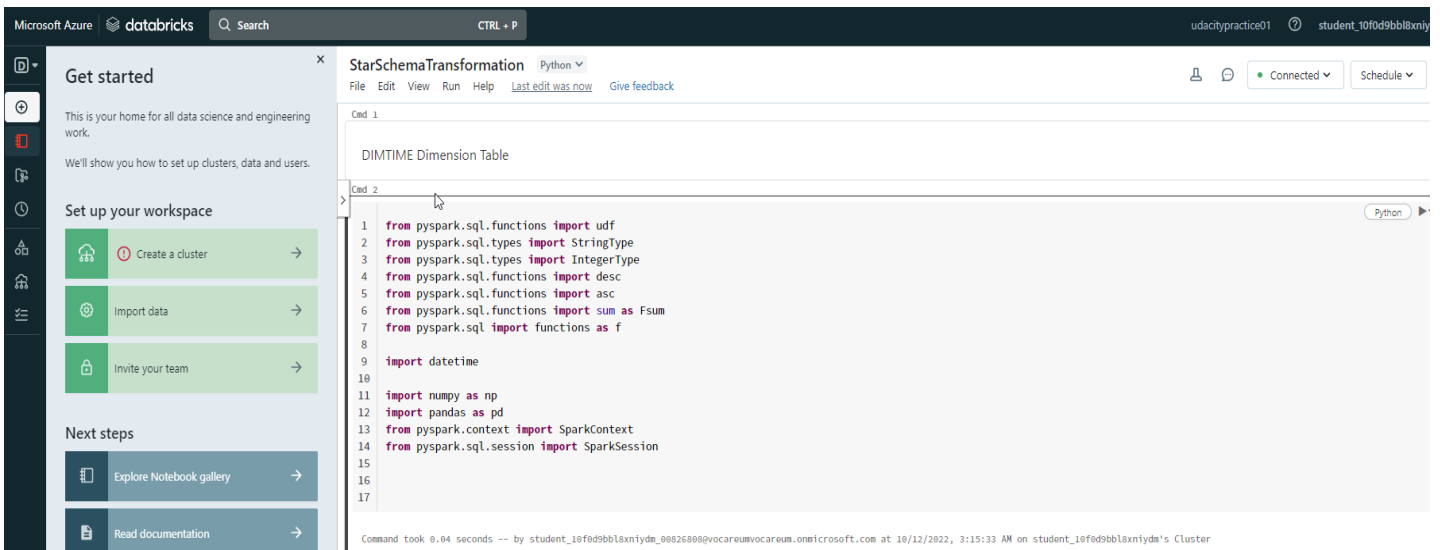
```
1 spark.sql("CREATE TABLE payments USING DELTA LOCATION '/delta/payments'")
2 spark.sql("CREATE TABLE riders USING DELTA LOCATION '/delta/riders'")
3 spark.sql("CREATE TABLE stations USING DELTA LOCATION '/delta/station'")
4 spark.sql("CREATE TABLE trips USING DELTA LOCATION '/delta/trip'")
```

Task 5: Transform



Using the Extract Python Notebook - StarSchemaTransformation, transformed from DBFS DELTA Storage to DBFS Database Tables

Using below dependencies created the fact and dimension tables



DIMTIME table:

Create dimtime table using dimtime dataframe in the databricks delta storage

Cmd 7

1

dimtime.write.format("delta").mode("overwrite").saveAsTable("dimtime")

▼ (4) Spark Jobs

▶ Job 11

View (Stages: 1/1)

▶ Job 13

View (Stages: 1/1)

▶ Job 14

View (Stages: 1/1, 1 skipped)

▶ Job 15

View (Stages: 1/1, 2 skipped)

Command took 3.74 seconds -- by student_10f0d9bbl8xniydm_00826808@vocareumvocareum.onmicrosoft.com at 10/12/2022, 4:23:28 AM on student_10f0d9bbl8xniydm's Cluster

Microsoft Azure

databricks

Search

CTRL + P

D

+

📄

🔗

🕒

🔧

🏠

📋

hive_metastore.default.dimtime

Refresh

student_10f0d9bbl8xniydm's Cluster

DetailsHistory

Description:

Created at: 2022-10-11 15:23:29

Last modified: 2022-10-11 15:23:30

Partition columns:

Number of files: 4

Size: 29.9 kB

Schema:

	col_name	data_type	comment
1	time_id	bigint	
2	adate	date	
3	ayear	bigint	
4	amonth	bigint	
5	aquarter	bigint	
6	adayofweek	bigint	
7	adayofmonth	bigint	


Sample Data:

	time_id	adate	ayear	amonth	aquarter	adayofweek	adayofmonth
1	1	2021-02-01	2021	2	1	0	1
2	2	2021-02-02	2021	2	1	1	2
3	3	2021-02-03	2021	2	1	2	3
4	4	2021-02-04	2021	2	1	3	4
5	5	2021-02-05	2021	2	1	4	5
6	6	2021-02-06	2021	2	1	5	6
7	7	2021-02-07	2021	2	1	6	7

1/3









DIMSTATION table:


Microsoft Azure


 databricks

Search

CTRL + P



hive_metastore.default.dimstation | 

student_10f0d9bbl8xnjydm's Cluster | 

Details

History

Description:

Created at: 2022-10-11 14:40:51

Last modified: 2022-10-11 15:19:59

Partition columns:

Number of files: 1

Size: 30.9 kB

Schema:

	col_name	data_type	comment
1	station_id	string	
2	stationname	string	
3	latitude	string	
4	longitude	string	
5			
6	# Partitioning		
7	Not partitioned		

Sample Data:

	station_id	stationname	latitude	longitude
1	525	Glenwood Ave & Touhy Ave	42.012701	-87.66605799999999
2	KA1503000012	Clark St & Lake St	41.88579466666667	-87.63110066666668
3	637	Wood St & Chicago Ave	41.895634	-87.672069
4	13216	State St & 33rd St	41.8347335	-87.6258275
5	18003	Fairbanks St & Superior St	41.89580766666667	-87.62025316666669
6	KP1705001026	LaSalle Dr & Huron St	41.894877	-87.632326
7	13253	Lincoln Ave & Waveland Ave	41.948797	-87.675278

DIMRIDER table:

Microsoft Azure | databricks

Search

CTRL + P

hive_metastore.default.dimrider | Refresh

student_10f0d9bb18xniydm's Cluster

Details History

Description:

Created at: 2022-10-11 15:28:35

Last modified: 2022-10-11 15:28:37

Partition columns:

Number of files: 2

Size: 2.11 MB

Schema:

	col_name	data_type	comment
1	rider_id	string	
2	firstname	string	
3	lastname	string	
4	address	string	
5	birthday	string	
6	ismember	string	
7			

Sample Data:

	rider_id	firstname	lastname	address	birthday	ismember
1	1000	Diana	Clark	1200 Alyssa Squares	1989-02-13	True
2	1001	Jennifer	Smith	397 Diana Ferry	1976-08-10	True
3	1002	Karen	Smith	644 Brittany Row Apt. 097	1998-08-10	True
4	1003	Bryan	Roberts	996 Dickerson Turnpike	1999-03-29	False
5	1004	Jesse	Middleton	7009 Nathan Expressway	1969-04-11	True
6	1005	Christine	Rodriguez	224 Washington Mills Apt. 467	1974-08-27	False
7	1006	Alicia	Taylor	1137 Angela Locks	2004-01-30	True

FACTPAYMENT table:

Microsoft AzuredatabricksSearchCTRL + P

hive_metastore.default.factpayment

student_10f0d9bbl8xnnydm's Cluster

Refresh

DetailsHistory

Description:

Created at: 2022-10-11 15:29:27

Last modified: 2022-10-11 15:29:29

Partition columns:

Number of files: 4

Size: 10.3 MB

Schema:

	col_name	data_type	comment
1	payment_id	string	
2	dates	string	
3	amount	string	
4	rider_id	string	
5			
6	# Partitioning		
7	Not partitioned		

Sample Data:

	payment_id	dates	amount	rider_id
1	1	2019-05-01	9.0	1000
2	2	2019-06-01	9.0	1000
3	3	2019-07-01	9.0	1000
4	4	2019-08-01	9.0	1000
5	5	2019-09-01	9.0	1000
6	6	2019-10-01	9.0	1000
7	7	2019-11-01	9.0	1000

FACTTRIP table:

Microsoft AzuredatabricksSearchCTRL + P

hive_metastore.default.facttrip

student_10f0d9bbl8xnnydm's Cluster

Refresh

DetailsHistory

Description:

Created at: 2022-10-11 15:42:41

Last modified: 2022-10-11 15:42:54

Partition columns:

Number of files: 4

Size: 179 MB

Schema:

	col_name	data_type	comment
1	trip_id	string	
2	rideable_type	string	
3	tripstarttime	string	
4	tripendtime	string	
5	start_station_id	string	
6	end_station_id	string	
7	duration	bigint	

Sample Data:

	trip_id	rideable_type	tripstarttime	tripendtime	start_station_id	end_station_id	duration	rider_age	rider_id
1	2228B8E5059252D7	classic_bike	2021-06-13 09:48:47	2021-06-13 10:07:23	KA1503000064	13021	0	30	34062
2	1826E16CB5486018	classic_bike	2021-06-21 22:59:13	2021-06-21 23:04:29	TA1306000010	13021	0	26	5342
3	3D986A0A5330B04D	classic_bike	2021-06-18 16:06:42	2021-06-18 16:12:02	TA1305000030	13021	0	26	3714
4	07E82F5E9C9E490F	classic_bike	2021-06-17 16:46:23	2021-06-17 17:02:45	TA1305000034	13021	0	18	18793
5	A8E948AECBFC2DD	docked_bike	2021-06-13 17:36:29	2021-06-13 18:30:39	TA1308000009	TA1308000009	0	28	43342
6	378F4AB323AA1D14	docked_bike	2021-06-13 13:20:10	2021-06-13 14:06:14	TA1308000009	TA1308000009	0	28	6693
7	38AD311DC2E81F8E	docked_bike	2021-06-16 17:14:30	2021-06-16 17:28:34	KA1503000019	KA1503000019	0	56	71480

All Fact and Dimension tables have been Created

The screenshot shows the 'Database Tables' interface in Databricks. At the top, there are tabs for 'Database Tables' and 'DBFS', and a 'Create Table' button. The interface is divided into three main sections: 'Catalogs', 'Databases', and 'Tables'. The 'Catalogs' section shows 'hive_metastore'. The 'Databases' section shows 'default'. The 'Tables' section lists several tables: 'dimrider', 'dimstation', 'dimtime', 'factpayment', 'facttrip', 'payments', 'riders', 'stations', and 'trips'. A mouse cursor is hovering over the 'dimstation' table.

Catalogs	Databases	Tables
hive_metastore	default	dimrider
		dimstation
		dimtime
		factpayment
		facttrip
		payments
		riders
		stations
		trips

NOTEBOOKS:

1. ExtractNotebook
2. StarSchemaTransformation

The screenshot shows the Databricks 'Workspace' interface. The top bar includes 'Microsoft Azure', the 'databricks' logo, a search bar, and a 'CTRL + P' shortcut. The left sidebar contains navigation options: 'Data Science & Engi...', 'Create', 'Workspace' (highlighted), 'Repos', 'Recents', 'Data', 'Compute', and 'Workflows'. The main area is titled 'Workspace' and shows a 'Shared' section with a list of notebooks: 'BusinessOutcomes', 'Data Wrangling in Databricks', 'Data Wrangling In Databricks Part 2', 'ExtractNotebook' (highlighted), 'Reading and Writing Data in Databricks', 'Reading and Writing Data Practice', and 'StarSchemaTransformation'.

Workspace	Shared
Shared	BusinessOutcomes
Users	Data Wrangling in Databricks
	Data Wrangling In Databricks Part 2
	ExtractNotebook
	Reading and Writing Data in Databricks
	Reading and Writing Data Practice
	StarSchemaTransformation