AWS - Database Workshop

Mehmet Kadri GOFRALILAR, 22301421

April 10, 2024

1 Workshop Steps

1.1 Lab 1 - Data Migration using AWS Data Migration Service

1.1.1 Environment Preparation

For the prerequisite part, I have pulled a db and filled PostgreSQL with that. Parent output is given in Google Drive.

1.1.2 Endpoint Creation

Afterwards, I created an Oracledb source endpoint, an Auroradb target endpoint and a DynamoDB target endpoint according to given parameters. Then I tested their connections, which are shown in Figure 1.

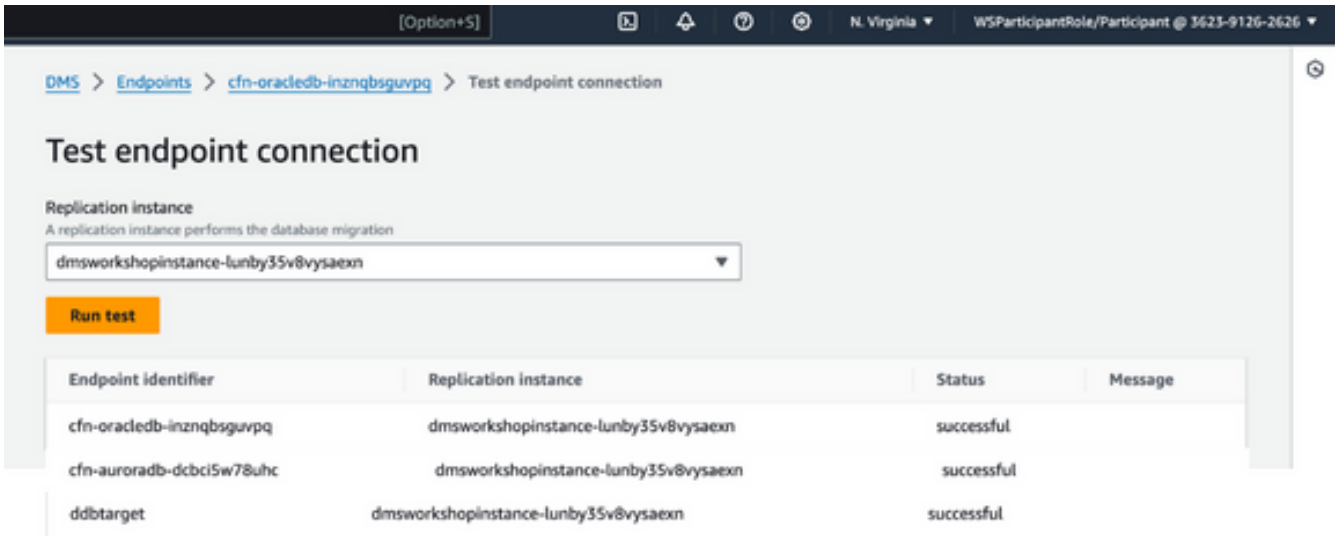


Figure 1: Test results of endpoint connections cropped and merged

1.1.3 Data Migration

Oracle ⇒ DynamoDB:
Database is first migrated from Oracle to DynamoDB. During this process, one of the default settings ("Turn on premigration assessment") caused trouble in the beginning, even though the tutorial did not mention any changes to it. Apparently it was not supposed to be selected, which is found out by trial & error.

Oracle ⇒ Aurora PostgreSQL
Same process went on for migration from Oracle to Aurora PostgreSQL. Same issue happened and solved.

1.1.4 Validation

Finally, we can see that the data migration is completed in Figure 2.

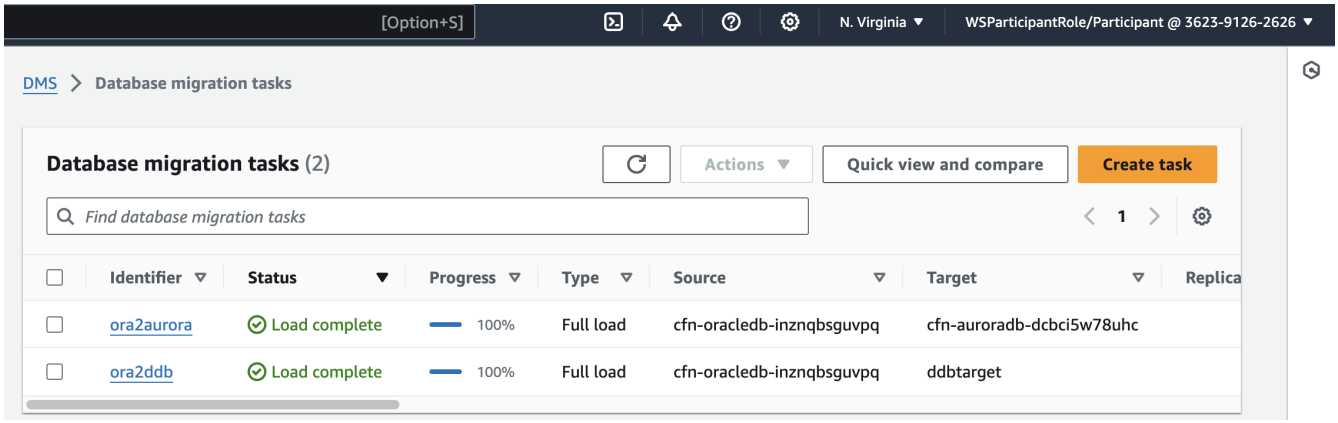


Figure 2: Completed migrations, which are from Oracle to Aurora and DynamoDB

1.2 Lab 2 - Data processing using Amazon DynamoDB and Amazon Aurora

1.2.1 Setup AWS Cloud 9 Environment

AWS SAM (Serverless Application Model) CLI is updated and Boto3 (AWS SDK for Python) is installed.

1.2.2 Enable Amazon DynamoDB Streams

The Amazon DynamoDB stream is enabled.

1.2.3 Deploy AWS Lambda Function for DynamoDB Stream Integration

The binaries for PG8000 (a Python interface to PostgreSQL) are downloaded and will be deployed as an AWS Lambda Layer. Then, a SAM template that contains the configuration for the Lambda function and the Lambda Layer is deployed. While the file "template-out.yaml" is being inspected, reading the description of the function was informative that stated "Process completed taxi trip information from Amazon DynamoDB Streams and publishes the information to the trips table in Amazon Aurora database". Also we can see that packages are uploaded to S3 bucket without any issues, given in Figure 3.

```
WSParticipantRole:~/environment/amazon-rds-purpose-built-workshop/src/ddb-stream-processor (master) $ aws s3 ls s3://$S3_BUCKETNAME
2024-04-08 03:29:13     725392 96677758b368e03f56050616a31d5387
2024-04-08 03:29:13     705691 d9f2ef545642b1797636e4df00386d01
```

Figure 3: Uploaded packages to S3

However, at this point, an error came up, shown in Figure 4. It seems like our database name is not defined.

```
WSParticipantRole:~/environment/amazon-rds-purpose-built-workshop/src/ddb-stream-processor (master) $ sam deploy --template-file template-out.yaml
--capabilities CAPABILITY_IAM --stack-name SAM-DDB-STREAM-APG --parameter-overrides LambdaLayerNameParameter=aws-db-workshop-pg8000-layer DDBStream
Name=$AWSDBWORKSHOP_DDB_STREAM_NAME SecurityGroupIds=$LAMBDASECURITYGROUP_ID VpcSubnetIds=$LAMBDASUBNET1_ID,$LAMBDASUBNET2_ID DatabaseName=$AURORAD
B_NAME DatabaseHostName=$AURORACLUSTERENDPOINT_NAME DatabaseUserName=$AURORADBMASTERUSER_NAME DatabasePassword=$PGPASSWORD
Usage: sam deploy [OPTIONS]
Try 'sam deploy -h' for help.

Error: Invalid value for '--parameter-overrides': DatabaseName= is not in valid format. It must look something like 'ParameterKey=KeyPairName,Param
eterValue=MyKey ParameterKey=InstanceType,ParameterValue=t1.micro' or 'KeyPairName=MyKey InstanceType=t1.micro'
WSParticipantRole:~/environment/amazon-rds-purpose-built-workshop/src/ddb-stream-processor (master) $ $AURORADB_NAME
WSParticipantRole:~/environment/amazon-rds-purpose-built-workshop/src/ddb-stream-processor (master) $ echo $AURORADB_NAME
```

Figure 4: Database Name Error

After backtracking, we see our mistake and fix it, given in Figure 5. Then, the issue is solved and the Lambda Function is deployed as we will see later.

```
WSParticipantRole:~/environment/amazon-rds-purpose-built-workshop/src/ddb-stream-processor (master) $ ©AURORADB_NAME=$(aws cloudformation describe-
stacks --stack-name $AWSDBWORKSHOP_CFSTACK_NAME | jq -r '.Stacks[].Outputs[] | select(.OutputKey=="AuroraDBName") | .OutputValue')

bash: ©AURORADB_NAME=taxidb: command not found
```

Figure 5: Error fixed

.

1.2.4 Deploy AWS Lambda Functions for Taxi Ride workflow

In this step, while inspecting the "template-out.yaml" file, we learn about 3 new functions and their aim.
RiderBookTripFunction:
Takes rider_id and rider_mobile as input and books a trip for the rider by updating ws-db-workshop-trips table in DynamoDB.
DriverAcceptTripFunction:
Takes rider_id and trip_info as input and accepts a trip by the driver by updating ws-db-workshop-trips table in DynamoDB.
DriverCompleteTripFunction:
Takes rider_id and trip_info as input and completes a trip by the driver by updating ws-db-workshop-trips table in DynamoDB.
Now, we can see these functions along the previous one, given in Figure 6.

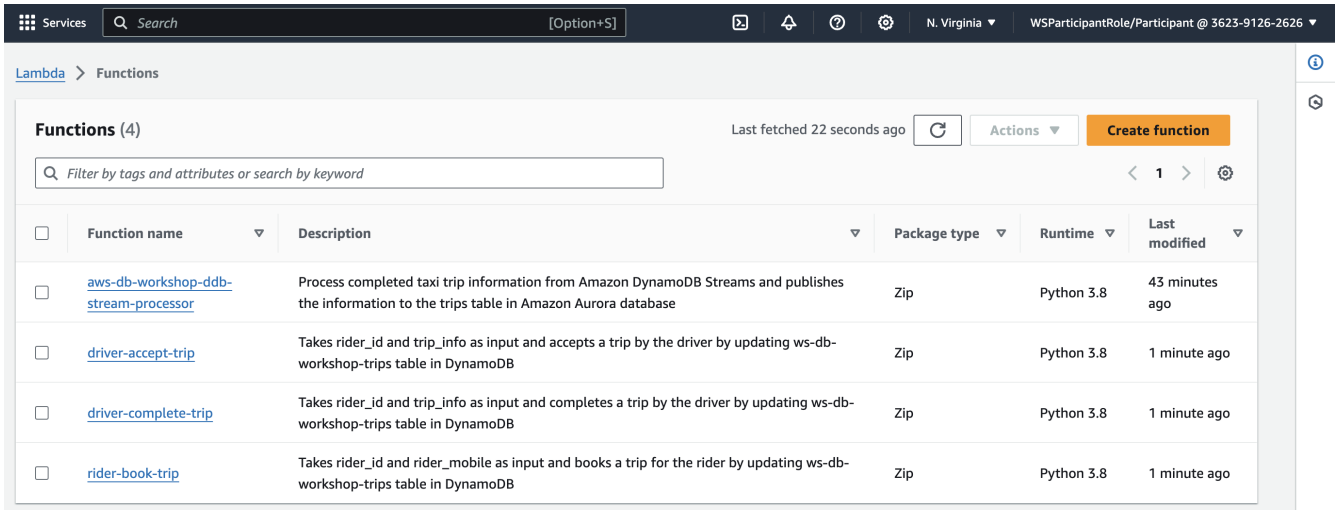


Figure 6: Deployed Functions

1.2.5 Create and Deploy API for Taxi Ride workflow

We learned how to create an API using Amazon API Gateway by defining its resources and methods (which were all GET methods in our case) and then, deployed it.

1.2.6 Taxi Ride Workflow

Later, we simulated a Taxi Ride Workflow using the API we deployed using the Invoke URLs of the methods in the following order:

1. riderbook given in Figure 7.
2. driveraccept given in Figure 8.
3. drivercomplete given in Figure 9.

Figure 7: riderbook

Figure 8: driveraccept

Figure 9: drivercomplete

Then, we reviewed the trip details as given below in Figure 10.

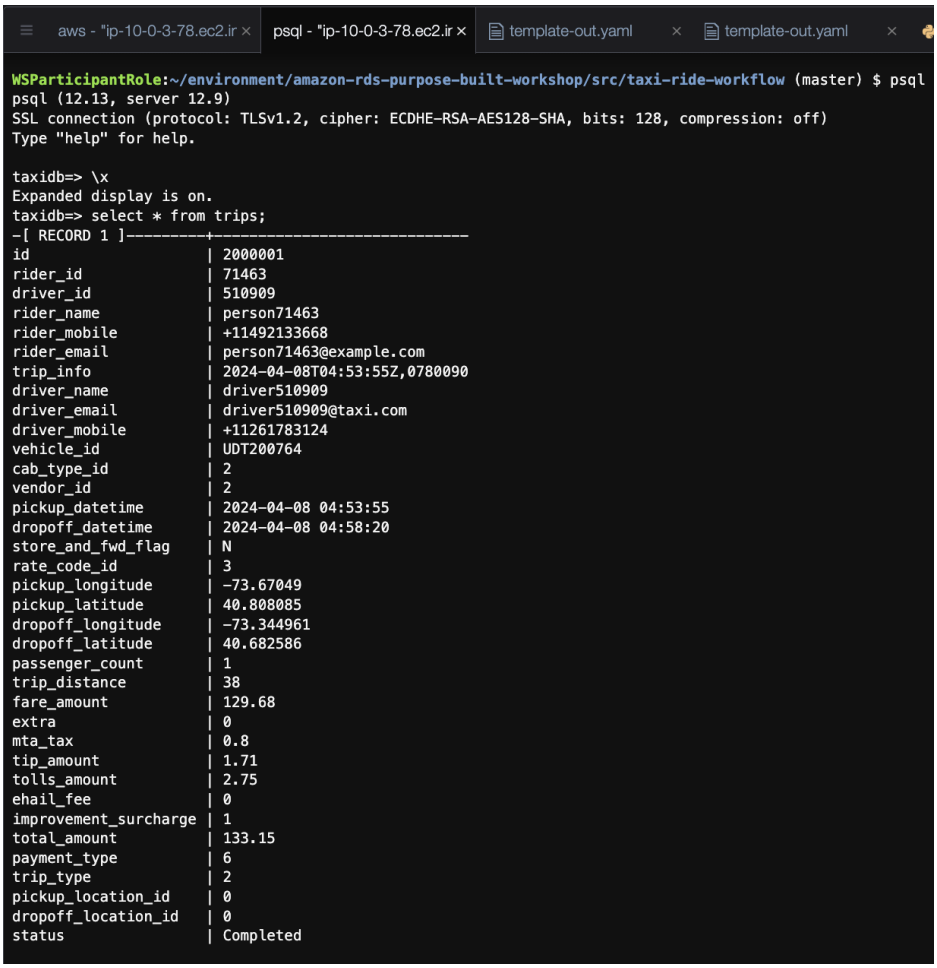


Figure 10: Trip Details

Finally, we called billingandpayments() and simulated the billing and payment workflow. Given output shows that the billing cycle is complete in Figure 11.

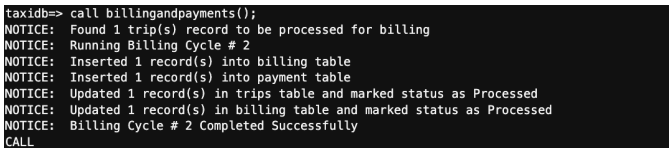


Figure 11: billingandpayments()

We can also see the billing is added to the billing table as given in Figure 12.

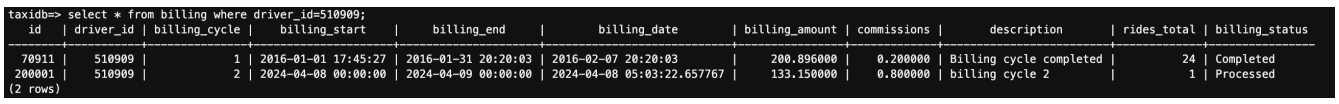


Figure 12: Billing Table

1.3 Lab 3 - Query multiple data sources using Amazon Athena federated query

1.3.1 Prepare the Environment

We made preparations to deploy Athena connectors.

1.3.2 Setup Athena Connectors and Catalogs

We deployed Amazon DynamoDB and and Aurora PostgreSQL data source connectors according to the given parameters and options, which we will query in the last step using Athena federated query. Data sources and added Lambda functions are given in Figure 13 and Figure 14, respectively.

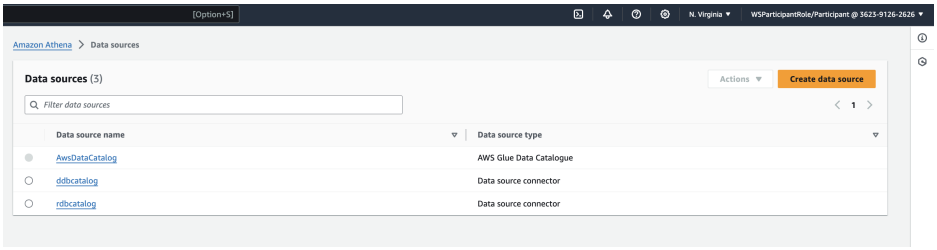


Figure 13: Created Data Sources

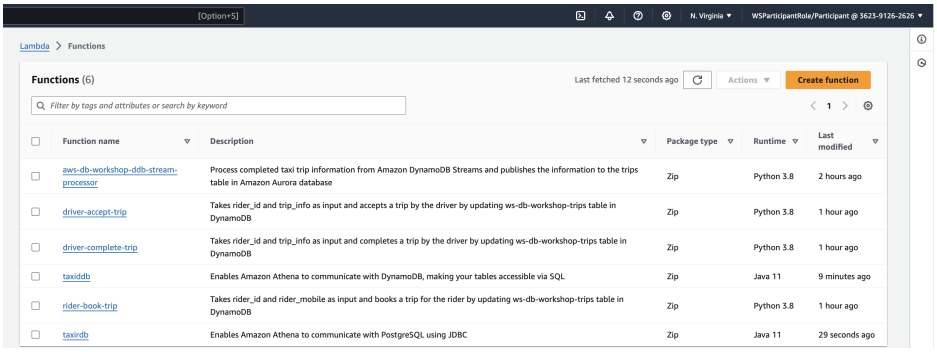
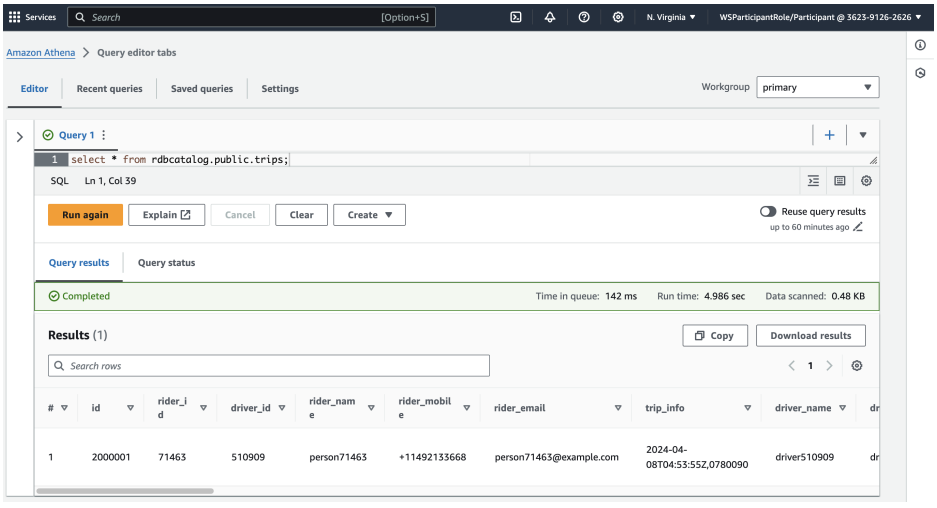
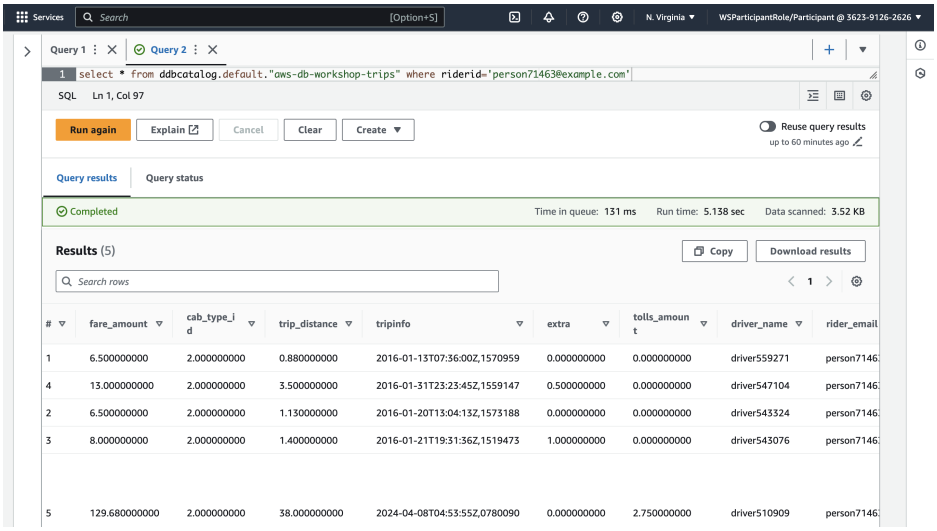


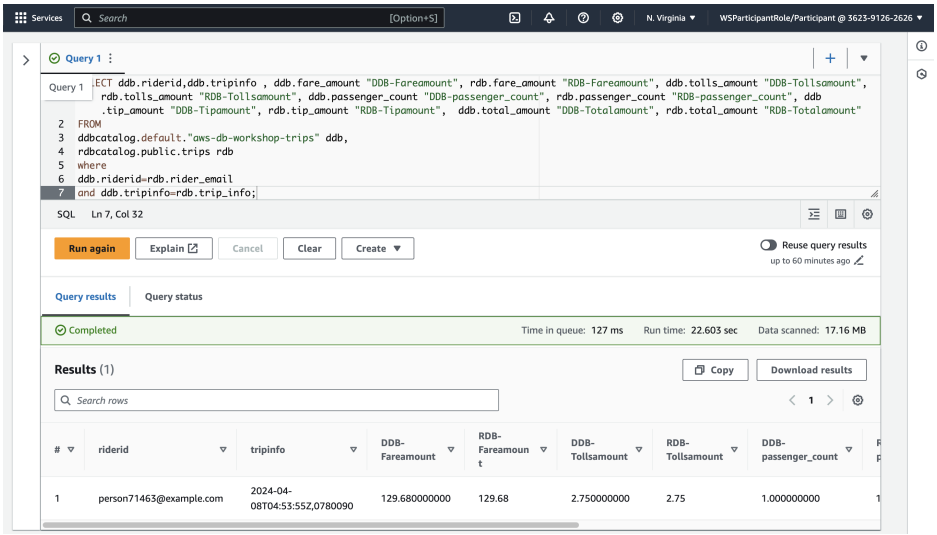
Figure 14: Lambda functions, including new ones

1.3.3 Query multiple data sources using Athena Federated Query

Sample queries are ran according to the instructions. Following figures show that the trip record is accurate.




Services
Search
[Option+S]
N. Virginia
WSParticipantRole/Participant @ 3623-9126-2626
Query 1
Query 2
1 select * from ddbcatalog.default."aws-db-workshop-trips" where riderid='person71463@example.com'
SQL Ln 1, Col 97
Run again
Explain
Cancel
Clear
Create
Reuse query results
up to 60 minutes ago
Query results
Query status
Completed
Time in queue: 131 ms
Run time: 5.138 sec
Data scanned: 3.52 KB
Results (5)
Copy
Download results
Search rows
1
fare_amount cab_type_id trip_distance tripinfo extra tolls_amount driver_name rider_email
1 6.500000000 2.000000000 0.880000000 2016-01-13T07:36:00Z,1570959 0.000000000 0.000000000 driver559271 person7146
4 13.000000000 2.000000000 3.500000000 2016-01-31T23:23:45Z,1559147 0.500000000 0.000000000 driver547104 person7146
2 6.500000000 2.000000000 1.130000000 2016-01-20T13:04:13Z,1573188 0.000000000 0.000000000 driver543324 person7146
3 8.000000000 2.000000000 1.400000000 2016-01-21T19:31:36Z,1519473 1.000000000 0.000000000 driver543076 person7146
5 129.680000000 2.000000000 38.000000000 2024-04-08T04:53:55Z,0780090 0.000000000 2.750000000 driver510909 person7146


Services
Search
[Option+S]
N. Virginia
WSParticipantRole/Participant @ 3623-9126-2626
Query 1
Query 1
ECT ddb.riderid,ddb.tripinfo , ddb.fare_amount "DDB-Fareamount", rdb.fare_amount "RDB-Fareamount", ddb.tolls_amount "DDB-Tollsamount",
rdb.tolls_amount "RDB-Tollsamount", ddb.passenger_count "DDB-passenger_count", rdb.passenger_count "RDB-passenger_count", ddb
.tip_amount "DDB-Tipamount", rdb.tip_amount "RDB-Tipamount", ddb.total_amount "DDB-Totalamount", rdb.total_amount "RDB-Totalamount"
2 FROM
3 ddbcatalog.default."aws-db-workshop-trips" ddb,
4 rdbcatalog.public.trips rdb
5 where
6 ddb.riderid=rdb.rider_email
7 and ddb.tripinfo=rdb.trip_info;
SQL Ln 7, Col 32
Run again
Explain
Cancel
Clear
Create
Reuse query results
up to 60 minutes ago
Query results
Query status
Completed
Time in queue: 127 ms
Run time: 22.603 sec
Data scanned: 17.16 MB
Results (1)
Copy
Download results
Search rows
1
riderid tripinfo DDB-Fareamount RDB-Fareamount DDB-Tollsamount RDB-Tollsamount DDB-passenger_count
1 person71463@example.com 2024-04-08T04:53:55Z,0780090 129.680000000 129.68 2.750000000 2.75 1.000000000