Lab 4: Working with Amazon Athena

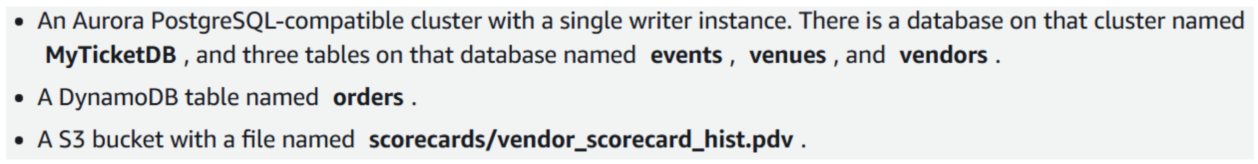
Objs-

1. Create databases and tables in Athena to query S3 objects.
2. Query data stored in S3 objects using Athena..
3. Query data from Aurora using Athena.
4. Query data from DynamoDB using Athena federated queries.
5. Access, join, and analyze data across Amazon S3, Aurora, and DynamoDB sources.

Athena features-

- Serverless: Analysts don't have to worry about managing servers.
- SQL Support: It allows running SQL queries on data from various sources like Amazon S3, DynamoDB, and relational databases.
- Data Integration: It can combine data from different sources to provide useful insights.

Pre-set:

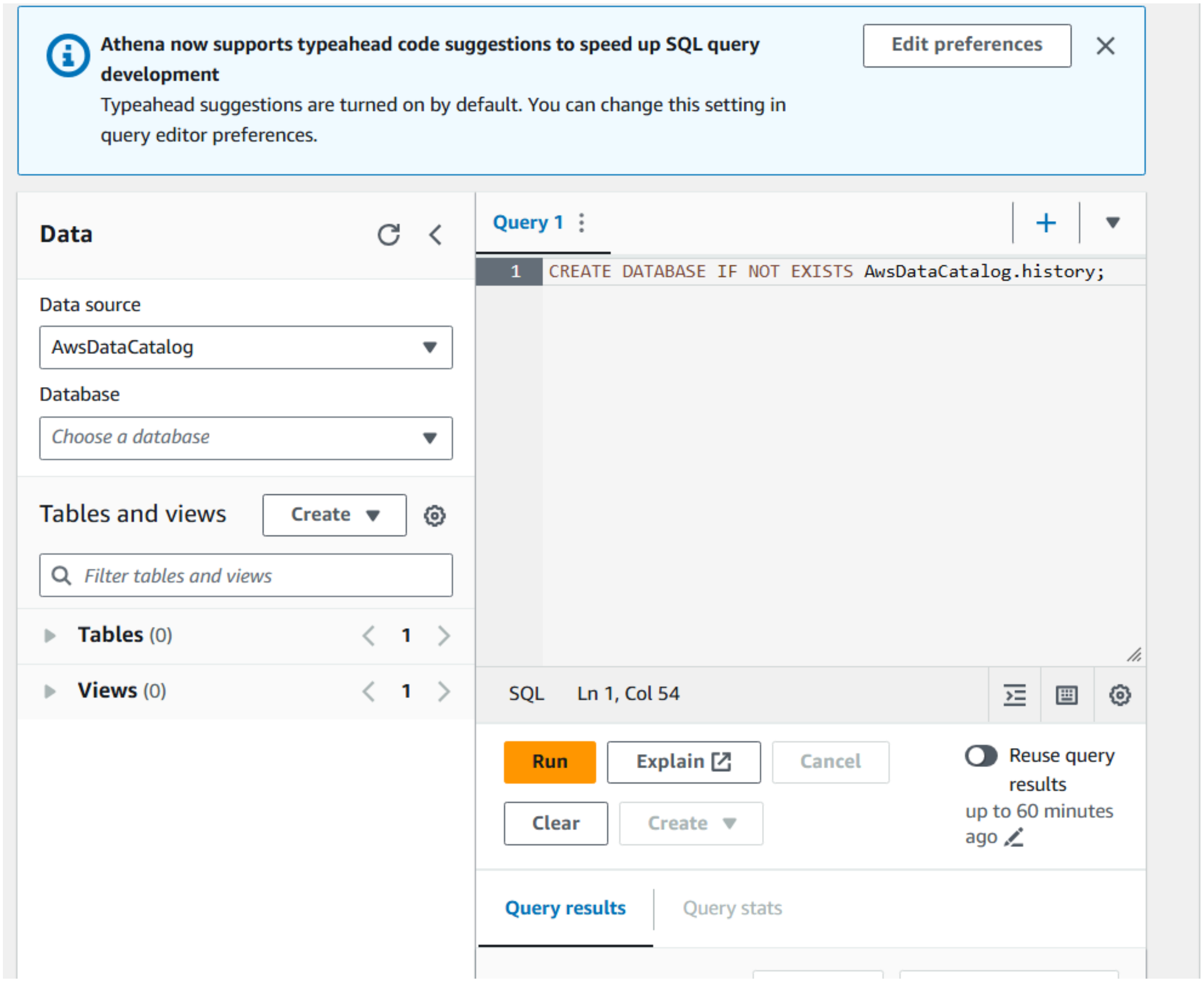


- An Aurora PostgreSQL-compatible cluster with a single writer instance. There is a database on that cluster named **MyTicketDB** , and three tables on that database named **events** , **venues** , and **vendors** .
- A DynamoDB table named **orders** .
- A S3 bucket with a file named **scorecards/vendor_scorecard_hist.pdv** .

Task 1 - create a database and table that uses a file stored in an S3
We also validate that the data is accessible via Athena

1.1 Open athena in console and launch query editor option.

1.2 Screenshot missed -
A pop-up will appear showing the path of S3 bucket, acknowledge it.
This location is also given in lab manual


Athena now supports typeahead code suggestions to speed up SQL query development
Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

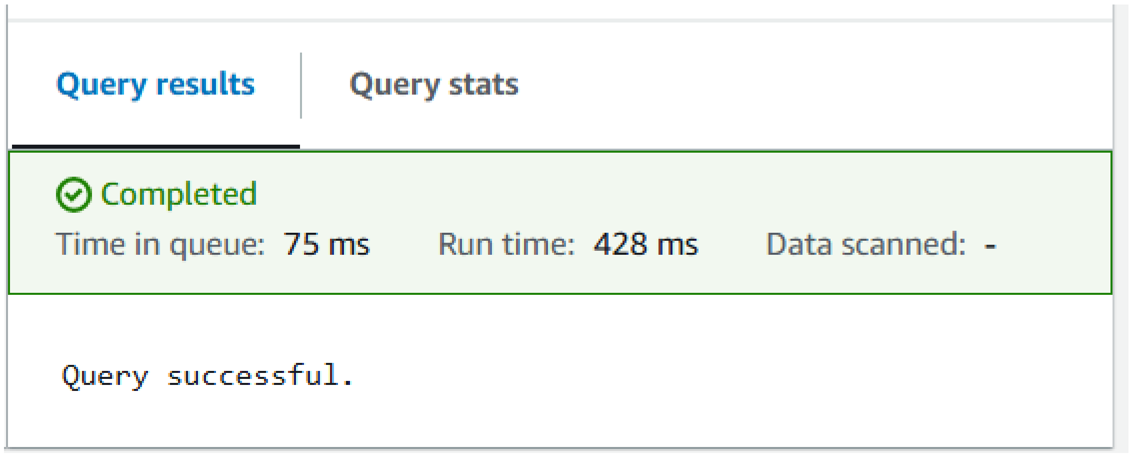
Data source: AwsDataCatalog
Database: Choose a database

```
CREATE DATABASE IF NOT EXISTS AwsDataCatalog.history;
```

SQL Ln 1, Col 54

1.3 run this query

CREATE DATABASE IF NOT EXISTS AwsDataCatalog.history;

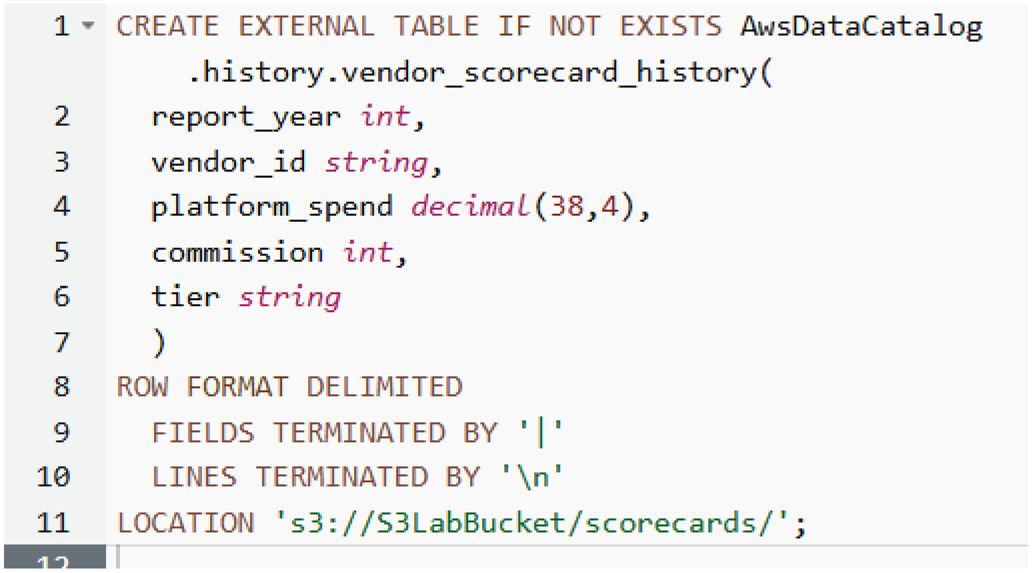

Query results | Query stats

Completed
Time in queue: 75 ms Run time: 428 ms Data scanned: -

```
Query successful.
```

1.4

1.5 replace bucket value and run this

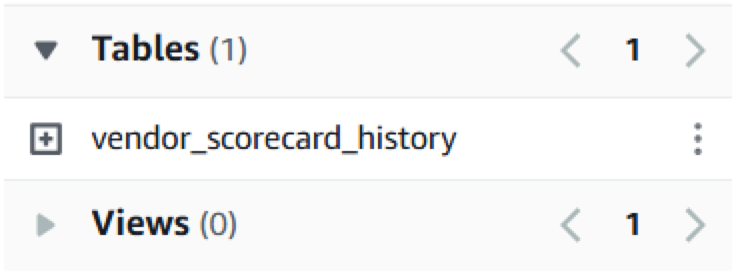


```
CREATE EXTERNAL TABLE IF NOT EXISTS AwsDataCatalog
    .history.vendor_scorecard_history(
  report_year int,
  vendor_id string,
  platform_spend decimal(38,4),
  commission int,
  tier string
  )
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '|'
  LINES TERMINATED BY '\n'
LOCATION 's3://S3LabBucket/scorecards/';
```

external table in Athena means the data isn't stored in Athena itself but remains in S3. Athena only stores the metadata (the schema).

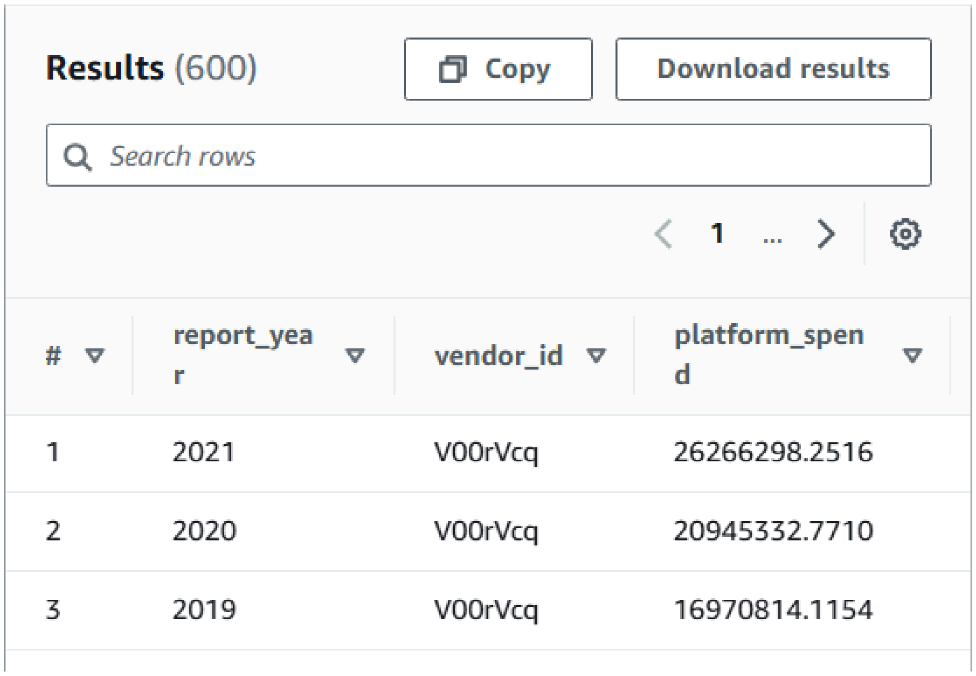
's3://S3LabBucket/scorecards/':
Specifies the location in Amazon S3 where the data files are stored. In this case, the data files are in the scorecards directory of the S3LabBucket bucket.

1.6 new table created



1.7 view the table using

SELECT * FROM AwsDataCatalog.history.vendor_scorecard_history;



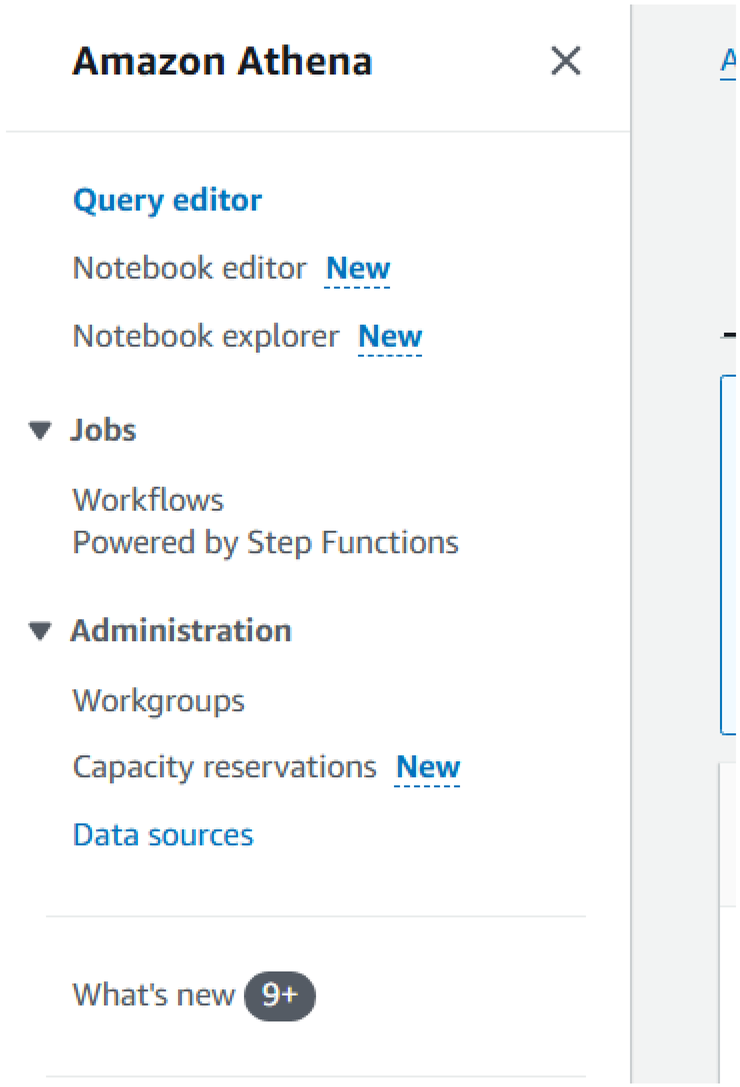
#	report_year	vendor_id	platform_spend
1	2021	V00rVcq	26266298.2516
2	2020	V00rVcq	20945332.7710
3	2019	V00rVcq	16970814.1154

Task 2: Setup Athena Federated Query access for the Event Management database

From aurora postgreSQL

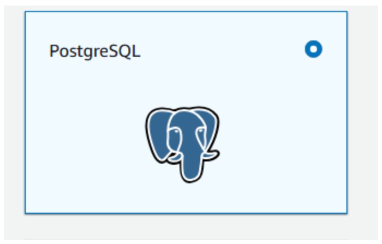
In this task, you create an Athena data source that connects to an the AnyCompany event management database, which consists of multiple tables in a Aurora PostgreSQL database.

2.1 click on data sources

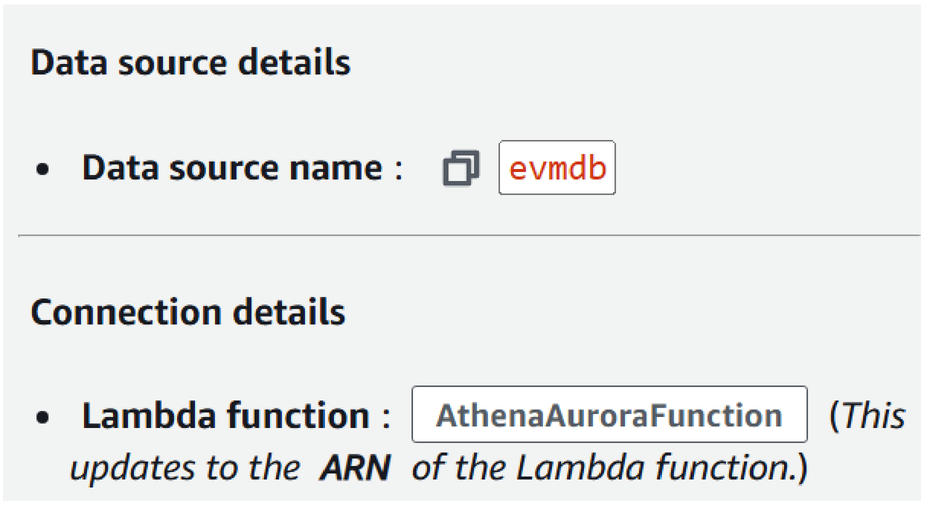


2.2 click on create data source

2.3 choose this



2.4 fill these data source details



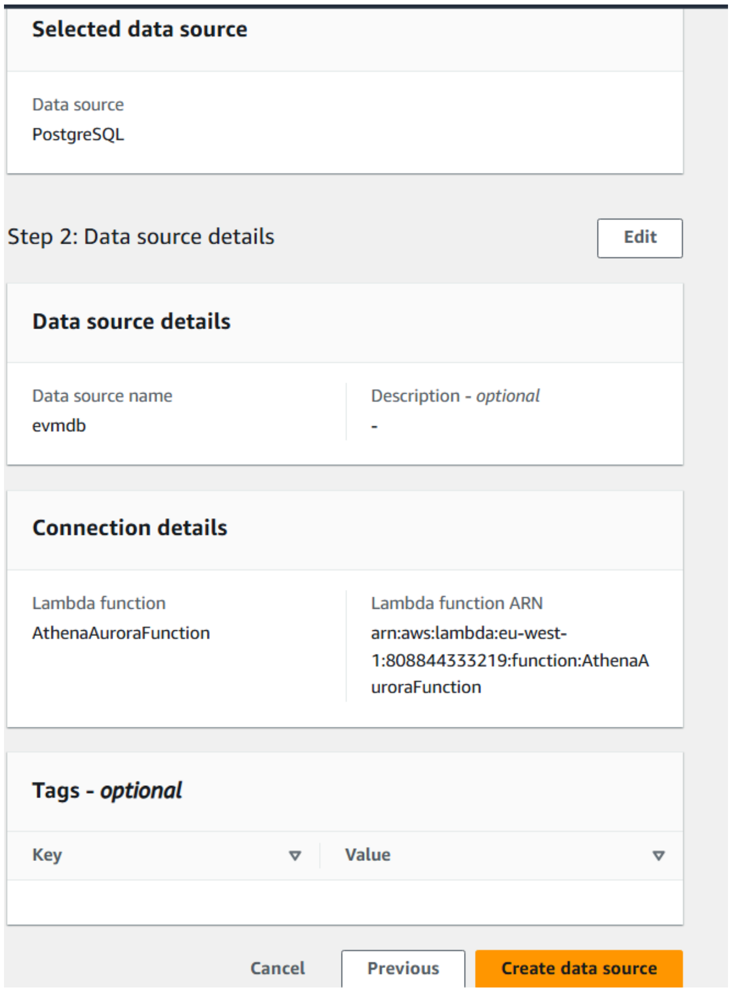
Data source details

- **Data source name :** evmdb

Connection details

- **Lambda function :** AthenaAuroraFunction *(This updates to the **ARN** of the Lambda function.)*

2.5 choose next



Selected data source

Data source
PostgreSQL

Step 2: Data source details

Data source details

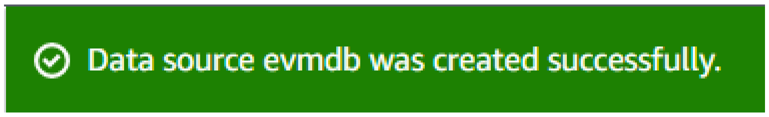
Data source name	Description - *optional*
evmdb	-

Connection details

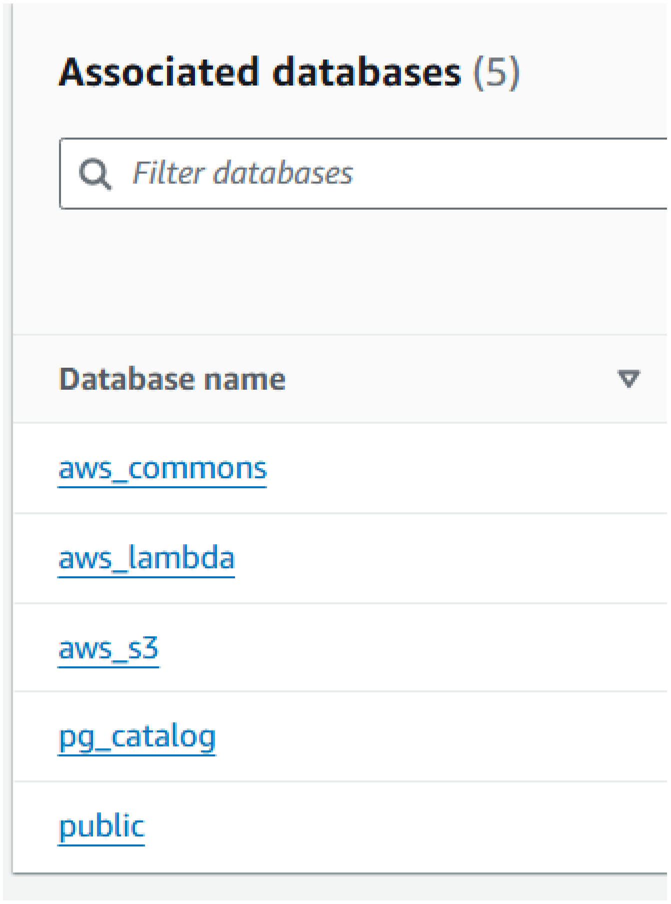
Lambda function	Lambda function ARN
AthenaAuroraFunction	arn:aws:lambda:eu-west-1:808844333219:function:AthenaAuroraFunction

Tags - *optional*

Key	Value

2.6 
Data source evmdb was created successfully.

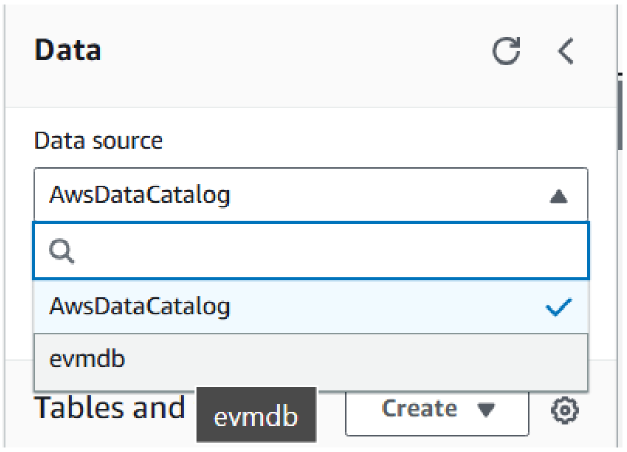
2.7 in the data source -


Associated databases (5)

Filter databases

Database name
aws_commons
aws_lambda
aws_s3
pg_catalog
public

2.8 return to query editor, and change data source


Data

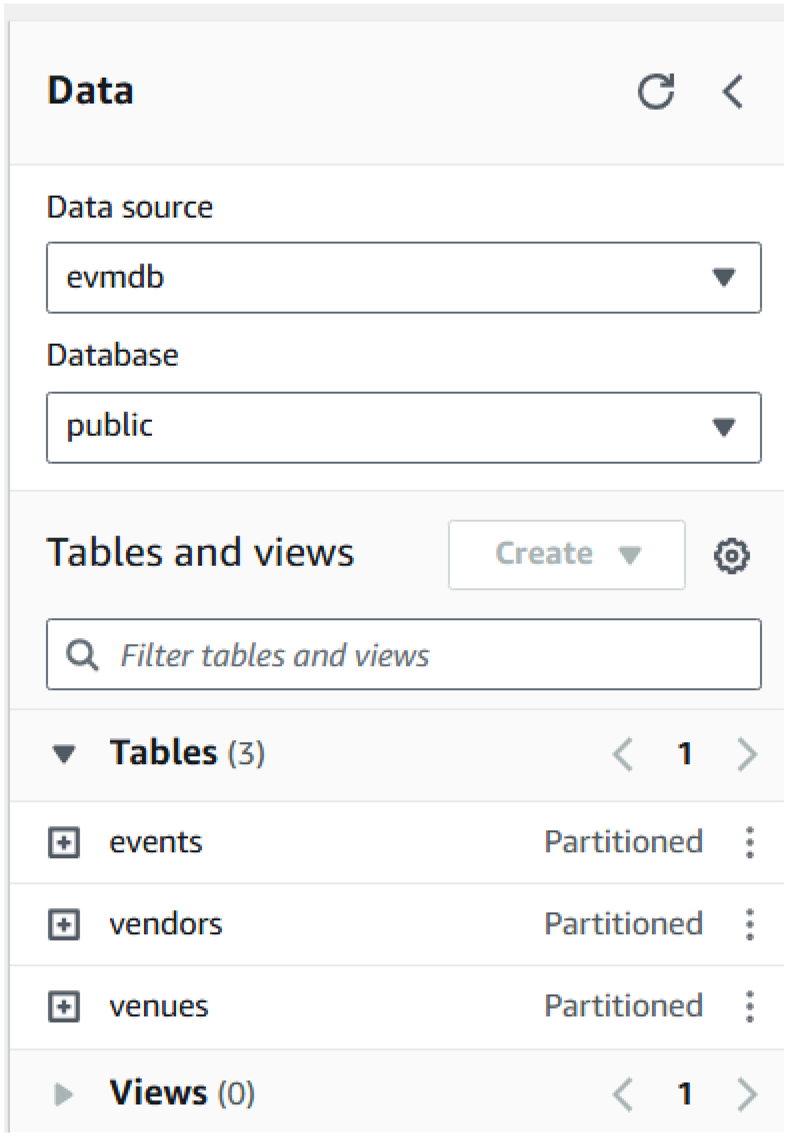
Data source

AwsDataCatalog

AwsDataCatalog ✓

evmdb

2.9 these tables show up in the public DB of this DS


Data

Data source: evmdb

Database: public

Tables and views

Tables (3)

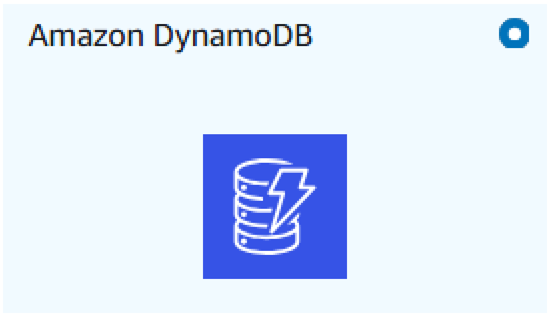
events	Partitioned
vendors	Partitioned
venues	Partitioned

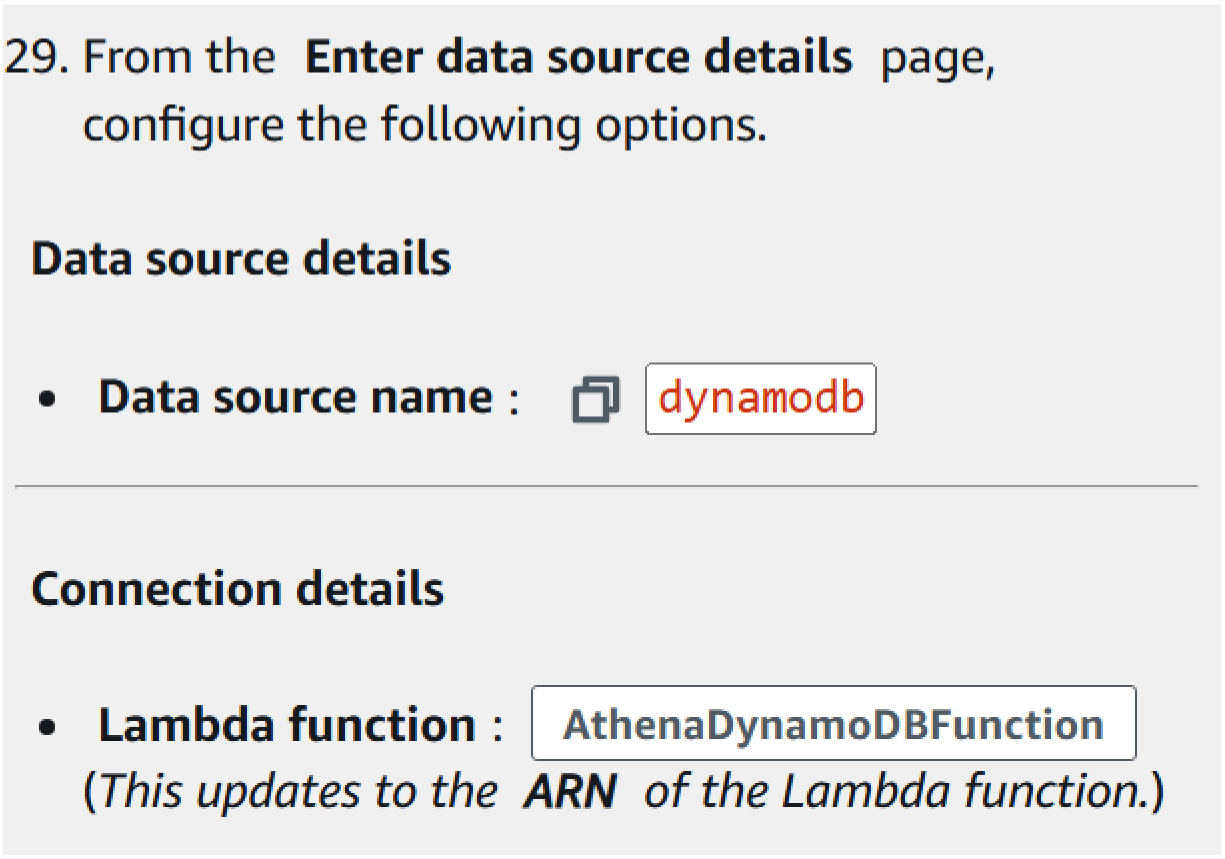
Views (0)

Task 3: Setup Athena Federated Query access for the Order Management table

From DynamoDB

3.1 repeat steps and create a new data source:





29. From the **Enter data source details** page, configure the following options.

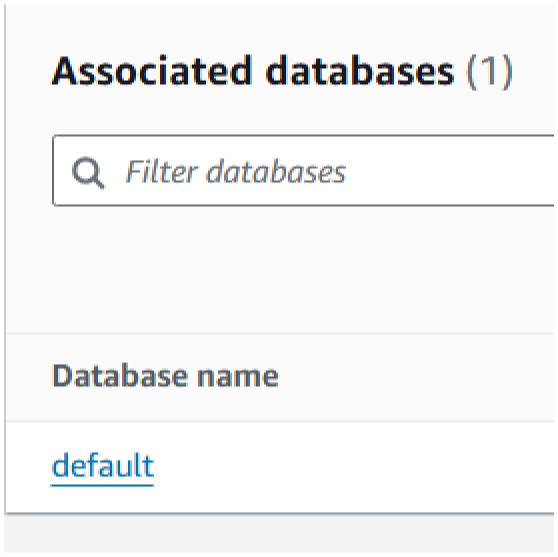
Data source details

- **Data source name :** dynamodb

Connection details

- **Lambda function :** AthenaDynamoDBFunction
  (*This updates to the **ARN** of the Lambda function.*)

3.2 databases in this

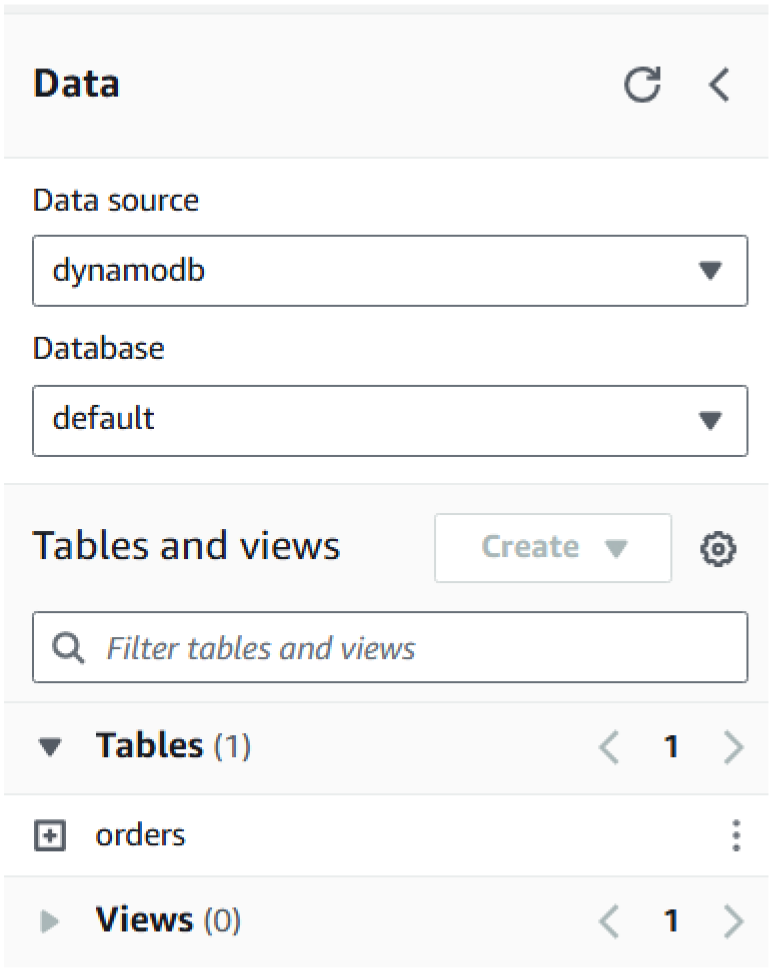


Associated databases (1)

Filter databases

Database name
default

3.3 orders table added



Data

Data source: dynamodb

Database: default

Tables and views

Tables (1)

orders

Views (0)

Task 4: Run SQL analytical query

4.1 run this command

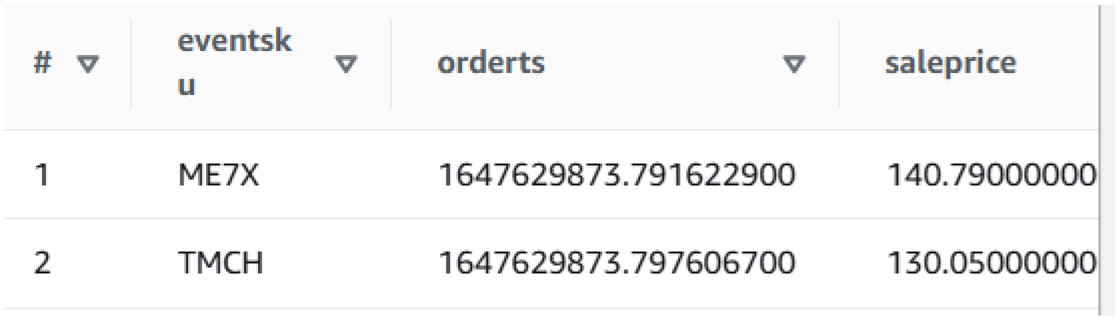
```
WITH
vendor_scorescard as (
select *,
case
when  report_year = 2021 then platform_spend
end as "spend(Y-1)",
case
when report_year = 2021 then commission
end as "comm(Y-1)",
case
when report_year = 2020 then platform_spend
end as "spend(Y-2)",
case
when report_year = 2020 then commission
end as "comm(Y-2)",
case
when report_year = 2019 then platform_spend
end as "spend(Y-3)",
case
when report_year = 2019 then commission
end as "comm(Y-3)"
from AwsDataCatalog.history.vendor_scorecard_history
where report_year between 2019 and 2021
),
vendor_scorescard_last_3y as (
select vendor_id,
        max("spend(Y-1)") as "spend(Y-1)",
        max("spend(Y-2)") as "spend(Y-2)",
        max("spend(Y-3)") as "spend(Y-3)",
        max("comm(Y-1)") as "comm(Y-1)",
        max("comm(Y-2)") as "comm(Y-2)",
        max("comm(Y-3)") as "comm(Y-3)"
from vendor_scorescard
group by vendor_id
),
current_year_spend as (
select
   vendors."vendor_id",
   coalesce(sum(orders."salePrice" * orders."orderQty"),0.00) as
"spend(CurrYear)",
   vendors.commission as "comm(CurrYear)" ,
   vendors."vendor_name" as "name",
   vendors."pterm" as "terms",
   vendors."status" as "status"
from
   ( evmdb.public.vendors as vendors
   left outer join evmdb.public.events as events ON (vendors."vendor_id"
= events."vendor_id") )
   left outer join dynamodb.default.orders as orders ON
(orders."eventSku" = events."event_sku")
group by vendors."vendor_id", vendors.commission,
vendors."vendor_name",
       vendors."pterm", vendors."status"
)
select
   curr."vendor_id",
   curr."spend(CurrYear)",
   vendors_hist."spend(Y-1)",
   vendors_hist."spend(Y-2)",
   vendors_hist."spend(Y-3)",
   curr."comm(CurrYear)",
   vendors_hist."comm(Y-1)",
   vendors_hist."comm(Y-2)",
   vendors_hist."comm(Y-3)",
   curr."name",
   curr."terms",
   curr."status"
from current_year_spend as curr
    left outer join vendor_scorescard_last_3y as vendors_hist ON
(vendors_hist."vendor_id" = curr."vendor_id");
```

This query combines historical spend and commission data from

1. AwsDataCatalog.history.vendor_scorecard_history,
2. evmdb.public.vendors, evmdb.public.events, and
3. dynamodb.default.orders

into a comprehensive report with details on spend, commission, vendor name, payment terms, and status.

Basically, it combines data from all three data sources, to run the query



#	eventsku	orderts	saleprice
1	ME7X	1647629873.791622900	140.79000000
2	TMCH	1647629873.797606700	130.05000000