

CIS 602: Big Data – Homework 2

Pradyoth Singenahalli Prabhu - 02071847

Task 1: Creating and querying an AWS Glue database and table in Athena

1) Creating database

```
CREATE DATABASE taxidata;
```

Successfully created database.

The screenshot shows the AWS Glue console interface. On the left, there is a navigation sidebar with links for Getting started, ETL jobs, Visual ETL, Notebooks, Job run monitoring, Data Catalog tables, Data connections, Workflows (orchestration), Data Catalog (expanded), Databases (selected), Tables, Stream schema registries, Schemas, Connections, Crawlers, Classifiers, Catalog settings, Data Integration and ETL, and Legacy pages. At the bottom of the sidebar, there are two buttons: 'Enable compact mode' (which is selected) and 'Enable new navigation'. The main content area is titled 'AWS Glue > Databases' and shows a table titled 'Databases (2)'. The table has columns for Name, Description, Location URI, and Created on (UTC). It lists two entries: 'default' (Created on September 27, 2023 at 21:21:11) and 'taxidata' (Created on September 27, 2023 at 21:21:12). There is also a search bar labeled 'Filter databases' and a button labeled 'Add database'.

2) Creating table named `yellow`

Skip to main content

Have you tried Amazon Athena for Apache Spark? With Athena for Spark, you can issue Spark SQL statements within Spark Python notebooks to create more complex insights. [Learn more](#) [Get started here](#).

If you have tried Athena Spark, let us know how we can make the feature better by providing us your feedback [here](#).

[Amazon Athena](#) > [Query editor](#) > Create table from S3 bucket data

Create table from S3 bucket data [Info](#)

Table details

Table name Table name must be from 3-128 characters and must be unique. Valid characters are a-z, A-Z, 0-9, _ (underscore). Table names tend to correspond to the directory where the data will be stored.

Description - optional Table description must be from 1-1024 characters. 1005 characters remaining.

Database configuration [Info](#)

Choose an existing database or create a new database Choose to access an existing database or to create a new database in order to create a new table. Athena stores the table schema in the AWS Glue Data Catalog.

Create a database Choose an existing database [taxdata](#)

Dataset [Info](#)

Location of input data set [View](#) [Browse S3](#) Input the path to the data set you want to process on Amazon S3. For example if your data is stored at s3://input-data-set/taxi/, please enter s3://input-data-set/taxi/. If your data is already partitioned, e.g. s3://input-data-set/taxi/year=2009/month=12/day=1 just input the base path s3://input-data-set/taxi/.

⚠️ Unable to verify if the selected bucket belongs to your current region. I acknowledge that using this location of input dataset will incur data transfer charges.

Data format [Info](#)

[Close](#) [Feedback](#)

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Data format [Info](#)

Table type File format SerDe library SerDe properties - optional [Add SerDe property](#)

Name Value [Remove](#)

Column details
Column name must be from 3-128 characters. Valid characters are a-z, A-Z, 0-9, _ (underscore). Certain advanced column types (mainly, structs) are not exposed in this interface.

Column name	Column type	Description - optional
vendor	string	Enter description Remove
pickup	timestamp	Enter description Remove
dropoff	timestamp	Enter description Remove
count	int	Enter description Remove
distance	int	Enter description Remove
ratecode	string	Enter description Remove
storeflag	string	Enter description Remove

[Close](#) [Feedback](#)

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Column name Column type Description - optional
tolls decimal Enter description Remove
Precision - optional Scale - optional 0

Column name Column type Description - optional
surcharge decimal Enter description Remove
Precision - optional Scale - optional 0

Column name Column type Description - optional
total decimal Enter description Remove
Precision - optional Scale - optional 0

Add a column Bulk add columns

Table properties - optional Info

Partition details - optional Info
Partitions are a way to group specific information together. Column name must be from 1-128 characters. Valid characters are a-z, A-Z, 0-9, _ (underscore). Certain advanced column types (namely, struct) are not exposed in this interface.

Bucketing - optional

Preview table query
The preview table query will be populated in the query editor. Creating tables allows you to be ready for real-time querying in the query editor.

```

13   extra decimal,
14   mta_tax decimal,
15   tip decimal,
16   tolls decimal,
17   surcharge decimal,
18   total decimal
19 ) COMMENT 'Table for taxi data'
20 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
21 WITH SERDEPROPERTIES ('field.delim' = ',')
22 STORE AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
23 LOCATION 's3://aws-tc-largeobjects/CUR-TF-200-ACDSCI-1/Lab2/yellow/'
24 TBLPROPERTIES ('classification' = 'csv');

```

Create table Cancel **Create table**

Creating a table named **yellow**.

```

CREATE EXTERNAL TABLE IF NOT EXISTS taxidata.yellow (
  `vendor` string,
  `pickup` timestamp,
  `dropoff` timestamp,
  `count` int,
  `distance` int,
  `ratecode` string,
  `storeflag` string,
  `pulocid` string,
  `dolocid` string,
  `paytype` string,
  `fare` decimal,
  `extra` decimal,
  `mta_tax` decimal,
  `tip` decimal,
  `tolls` decimal,
  `surcharge` decimal,
  `total` decimal
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = ',',
  'field.delim' = ','
) LOCATION 's3://aws-tc-largeobjects/CUR-TF-200-ACDSCI-1/Lab2/yellow/'
TBLPROPERTIES ('has_encrypted_data'='false');

```

The screenshot shows the AWS Athena Query Editor interface. On the left, the 'Data' sidebar displays the 'Data source' as 'AwsDataCatalog' and the 'Database' as 'taxidata'. Below this, the 'Tables and views' section shows a single table named 'yellow' with 11 columns: vendor, pickup, dropoff, count, distance, ratecode, storeflag, pulocid, dolocid, paytype, fare, extra, mta_tax, tip, tolls, and surcharge. The 'Views (0)' section is empty. The main workspace contains the following SQL code:

```

11   'paytype' string,
12   'fare' decimal,
13   'extra' decimal,
14   'mta_tax' decimal,
15   'tip' decimal,
16   'tolls' decimal,
17   'surcharge' decimal,
18   'total' decimal
19 ) COMMENT "Table for taxi data"
20 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
21 WITH SERDEPROPERTIES ('field.delim' = ',')
22 STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
23 LOCATION 's3://aws-tc-longobjects/CUR-TF-200-ACDSCI-1/Lab2/yellow/'
24 TBLPROPERTIES ('classification' = 'csv');

```

The 'Run again' button is highlighted in orange. The 'Query results' tab shows a green status bar indicating 'Completed' with a timestamp of '2023-05-03T14:44:44Z'. Below it, a message says 'Query successful.'.

Querying to preview data:

```
SELECT * FROM "AwsDataCatalog"."taxidata"."yellow" limit 10;
```

The screenshot shows the AWS Athena Query Editor interface after running the previous query. A blue info icon at the top left indicates that 'Athena now supports typeahead code suggestions to speed up SQL query development'. The main workspace shows the executed query:

```
1 SELECT * FROM "AwsDataCatalog"."taxidata"."yellow" limit 10;
```

The 'Run again' button is highlighted in orange. The 'Query results' tab shows a green status bar indicating 'Completed' with a timestamp of '2023-05-03T14:44:44Z'. Below it, a message says 'Query successful.'.

The results table is titled 'Results (10)' and contains 10 rows of data. The columns are: #, vendor, pickup, dropoff, count, distance, ratecode, storeflag, pulocid, dolocid, paytype, fare, extra, and mta_tax. The data is as follows:

#	vendor	pickup	dropoff	count	distance	ratecode	storeflag	pulocid	dolocid	paytype	fare	extra	mta_tax
1	1	2017-05-03 22:52:41.000	2017-05-03 23:01:38.000	1	2	1	N	48	107	1	9	1	
2	2	2017-05-03 22:52:41.000	2017-05-03 23:00:07.000	4	1	1	N	230	164	1	7	1	
3	2	2017-05-03 22:52:41.000	2017-05-03 22:59:45.000	2	0	1	N	68	48	1	7	1	
4	2	2017-05-03 22:52:41.000	2017-05-03 22:59:09.000	5	1	1	N	249	13	1	8	1	
5	1	2017-05-03 22:52:42.000	2017-05-03 23:03:19.000	1	2	1	N	230	239	1	10	1	
6	1	2017-05-03 22:52:42.000	2017-05-03 22:57:05.000	1	0	1	N	107	79	2	5	1	
7	1	2017-05-03 22:52:42.000	2017-05-03 22:57:01.000	2	0	1	N	237	237	2	5	1	
8	1	2017-05-03 22:52:42.000	2017-05-03 23:05:04.000	1	2	1	N	234	163	2	11	1	
9	1	2017-05-03 22:52:42.000	2017-05-03 22:57:01.000	1	0	1	N	186	164	1	5	1	
10	1	2017-05-03 22:52:42.000	2017-05-03 23:29:33.000	1	8	1	N	161	188	1	30	1	

Task 2: Optimizing Athena queries by using buckets

Creating a table named `jan` for the January 2017 data.

```
CREATE EXTERNAL TABLE IF NOT EXISTS jan (
    `vendor` string,
    `pickup` timestamp,
    `dropoff` timestamp,
    `count` int,
    `distance` int,
    `ratecode` string,
    `storeflag` string,
    `pulocid` string,
    `dolocid` string,
    `paytype` string,
    `fare` decimal,
    `extra` decimal,
    `mta_tax` decimal,
    `tip` decimal,
    `tolls` decimal,
    `surcharge` decimal,
    `total` decimal
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
    'serialization.format' = ',',
    'field.delim' = ','
)
LOCATION 's3://aws-tc-largeobjects/CUR-TF-200-ACDSCI-1/Lab2/January2017/'
TBLPROPERTIES ('has_encrypted_data'='false');
```

The message *Query successful* displays. Athena created a new table named `jan`, which contains the taxi data for only the month of January 2017. The table is listed in the **Tables** section on the left.

The screenshot shows the AWS Athena console interface. In the top navigation bar, 'Services' is selected. The main area displays the creation of a new table named 'jan'. The 'Data' tab is active, showing the table's schema and a preview of its data. The 'Tables' section lists the newly created 'jan' table. The 'Query results' section at the bottom shows the query was completed successfully. The bottom right corner includes links for 'cloudShell', 'Feedback', and copyright information.

2) Querying data:

Running the following query on the *yellow* table, which has data for the entire year. The data is not divided into monthly buckets.

```
SELECT count (count) AS "Number of trips" ,  
       sum (total) AS "Total fares" ,  
       pickup AS "Trip date"  
  FROM yellow WHERE pickup  
  
  between TIMESTAMP '2017-01-01 00:00:00'  
        and TIMESTAMP '2017-02-01 00:00:01'  
 GROUP BY pickup;
```

The screenshot shows the AWS CloudShell interface. On the left, the 'Data' sidebar is open, showing 'Data source: AwsDataCatalog' and 'Database: taxidata'. Below it, 'Tables and views' list 'jan' and 'yellow'. The main area contains a SQL editor with the following query:

```
1 SELECT count (count) AS "Number of trips" ,  
2       sum (total) AS "Total fares" ,  
3       pickup AS "Trip date"  
4  FROM yellow WHERE pickup  
5  
6  between TIMESTAMP '2017-01-01 00:00:00'  
7  and TIMESTAMP '2017-02-01 00:00:01'  
8 GROUP BY pickup;
```

Below the editor, the status bar indicates: Time in queue: 148 ms, Run time: 11.515 sec, Data scanned: 9.32 GB. The results table shows the first 10 rows of data:

#	Number of trips	Total fares	Trip date
1	6	69	2017-01-05 16:31:55.000
2	5	75	2017-01-17 21:34:03.000
3	9	134	2017-01-17 21:38:23.000
4	6	226	2017-01-17 21:38:26.000
5	2	45	2017-01-17 21:39:10.000
6	7	92	2017-01-17 21:45:16.000
7	2	16	2017-01-17 21:47:01.000
8	5	40	2017-01-17 21:47:26.000
9	3	33	2017-01-17 21:50:27.000
10	5	97	2017-01-17 21:53:49.000

Time in queue: 148 ms

Run time: 11.515 sec

Data scanned: 9.32 GB

Running the following query on the `jan` table:

```
SELECT count (count) AS "Number of trips" ,  
       sum (total) AS "Total fares" ,  
       pickup AS "Trip date"  
  FROM jan  
 GROUP BY pickup;
```

The screenshot shows the AWS Athena console interface. On the left, there's a sidebar with 'Data' selected, showing 'AwsDataCatalog' as the data source and 'taxidata' as the database. Below that are sections for 'Tables and views' (jan, yellow) and 'Views (0)'. The main area is titled 'Query 6' and contains the following SQL code:

```

1 SELECT count(*) AS "Number of trips",
2       sum (total) AS "Total fares",
3       pickup AS "Trip date"
4 FROM jan
5 GROUP BY pickup;

```

Below the code, it says 'SQL Ln 4, Col 9'. There are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. A note says 'Reuse query results up to 60 minutes ago'. The 'Query results' tab is selected, showing a green status bar with 'Completed' and performance metrics: 'Time in queue: 190 ms', 'Run time: 12.065 sec', and 'Data scanned: 815.30 MB'. The results table has a header row with '#', 'Number of trips', 'Total fares', and 'Trip date'. The data follows:

#	Number of trips	Total fares	Trip date
1	7	175	2017-01-05 21:02:56.000
2	4	83	2017-01-05 21:02:48.000
3	1	35	2017-01-05 21:03:07.000
4	6	60	2017-01-05 21:03:17.000
5	1	12	2017-01-05 21:03:56.000
6	4	131	2017-01-05 21:04:11.000
7	9	236	2017-01-05 21:04:17.000
8	8	100	2017-01-05 21:04:18.000
9	8	149	2017-01-05 21:04:30.000
10	7	122	2017-01-05 21:05:11.000
11	2	66	2017-01-05 21:05:18.000
12	5	49	2017-01-05 21:05:19.000

At the bottom, there are links for 'cloudShell', 'Feedback', and copyright information: '© 2023, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

Time in queue: 190 ms

Run time: 12.065 sec

Data scanned: 815.30 MB

Task 3: Optimizing Athena queries by using partitions

Creating a new table called `taxidata.creditcard` that is partitioned for `paytype = 1`

```

CREATE TABLE taxidata.creditcard
WITH (
  format = 'PARQUET'
)AS
SELECT * from "yellow"
WHERE paytype = '1';

```

The screenshot shows the Amazon Athena Query Editor. At the top, there's a banner indicating that Athena now supports typeahead code suggestions. The main area has tabs for 'Data' and 'Tables and views'. Under 'Tables and views', there are three tables listed: 'creditcard', 'jan', and 'yellow'. The 'yellow' table is currently selected. The SQL editor contains the following query:

```
1 CREATE TABLE taxidata.creditcard
2 - WITH (
3   format = 'PARQUET'
4   ) AS
5   SELECT * From "yellow"
6   WHERE paytype = '1';
7
```

The status bar at the bottom indicates a 'Completed' query with a time in queue of 174 ms, a run time of 24.372 sec, and data scanned of 9.32 GB.

Time in queue: 174 ms

Run time: 24.372 sec

Data scanned: 9.32 GB

Querying the non-partitioned data in the `yellow` table:

```
SELECT sum (total), paytype FROM yellow
WHERE paytype = '1' GROUP BY paytype;
```

The screenshot shows the Amazon Athena Query Editor interface. At the top, there's a navigation bar with 'aws' logo, 'Services', 'Search', and a user dropdown. Below it, the title 'Amazon Athena > Query editor' and tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Settings'. A workgroup dropdown is set to 'primary'. A message box at the top left says '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.' with a 'Edit preferences' button.

The main area has a 'Data' sidebar on the left with 'Data source' set to 'AwsDataCatalog' and 'Database' set to 'taxidata'. Under 'Tables and views', there are three tables: 'creditcard', 'jan', and 'yellow', and zero views. The 'creditcard' table is expanded. The main workspace shows a query editor with the following SQL:

```
1 SELECT sum (total), paytype FROM yellow
2 WHERE paytype = '1' GROUP BY paytype;
```

The status bar at the bottom indicates: 'Time in queue: 177 ms', 'Run time: 7.967 sec', and 'Data scanned: 9.32 GB'. There are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. A 'Query results' tab is selected, showing a single row of results:

#	_col0	paytype
1	1351232654	1

At the bottom of the page, there are links for 'cloudShell', 'Feedback', and copyright information: '© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

Time in queue: 177 ms

Run time: 7.967 sec

Data scanned: 9.32 GB

Querying the partitioned data in the `creditcard` table:

```
SELECT sum (total), paytype FROM creditcard
WHERE paytype = '1' GROUP BY paytype;
```

The screenshot shows the Amazon Athena Query Editor interface. The top navigation bar includes the AWS logo, Services, Search, and a user profile. The main area has tabs for Editor, Recent queries, Saved queries, and Settings, with Workgroup set to Primary. A message at the top states "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." Below this is a toolbar with buttons for Create Database, Create Table, and Create View, along with options to Run again, Explain, Cancel, Clear, and Create. The SQL editor shows the following query:

```

1 SELECT sum(total), paytype FROM creditcard
2 WHERE paytype = '1' GROUP BY paytype;
3

```

The results section shows a completed query with the following details:

- Completed
- Time in queue: 182 ms
- Run time: 1.936 sec
- Data scanned: 73.02 MB

The results table displays one row:

#	_col0	paytype
1	1351232654	1

Time in queue: 182 ms

Run time: 1.936 sec

Data scanned: 73.02 MB

Task 4: Using Athena views

Creating a view called `cctrrips` for the total dollar value of fares that were paid with a credit card:

```

CREATE VIEW cctrrips AS
  SELECT "sum"("fare") "CreditCardFares"
    FROM yellow
   WHERE ("paytype"='1');

```

The screenshot shows the AWS Athena Query Editor interface. The top navigation bar includes the AWS logo, 'Services' dropdown, a search bar, and account information ('N. Virginia' and 'vocabs/user2486983=psingenhalliprabhu@umassd.edu @ 4284-441...'). The main area has tabs for 'Editor' (selected), 'Recent queries', 'Saved queries', and 'Settings'. A message at the top states: '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.' Below this is a 'Data' sidebar with 'Data source' set to 'AwsDataCatalog' and 'Database' set to 'taxidata'. The 'Tables and views' section lists tables like 'creditcard', 'jan', 'yellow', and views like 'cctrips' and 'creditcardfares'. The central workspace contains a query history and a new query entry. The query history shows several completed queries numbered 1 through 9. The new query entry is: '1 CREATE VIEW cctrips AS 2 SELECT "sum"(fare) "CreditCardFares" 3 FROM yellow 4 WHERE ("paytype"='1');'. The status bar indicates the query is at 'Ln 5, Col 1'. Below the workspace are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. To the right, there's a 'Reuse query results' link. The bottom of the screen shows a footer with links for 'cloudShell', 'Feedback', and copyright information: '© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

Time in queue: 169 ms

Run time: 848 sec

Create a view called `cashtrips` for the total dollar value of fares that were paid with cash:

```
CREATE VIEW cashtrips AS
  SELECT "sum"(fare) "CashFares"
    FROM yellow
   WHERE ("paytype"='2');
```

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

Data source: AwsDataCatalog
Database: taxidata

Tables and views: creditcard, jan, yellow, cashtrips, cashfares, cctrips

SQL:

```

1 CREATE VIEW cashtrips AS
2   SELECT "sum"(fare) "CashFares"
3   FROM yellow
4   WHERE ("paytype"='2');
5

```

Run again Explain Cancel Clear Create

Query results: Completed Time in queue: 147 ms Run time: 604 ms Data scanned: -

Query successful.

Time in queue: 147 ms

Run time: 604 sec

Select all the records from the view `cctrips`:

```
Select * from cctrips;
```

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

Data source: AwsDataCatalog
Database: taxidata

Tables and views: creditcard, jan, yellow, cashtrips, cashfares, cctrips

SQL:

```

1 Select * from cctrips;

```

Run again Explain Cancel Clear Create

Query results: Completed Time in queue: 362 ms Run time: 7.944 sec Data scanned: 9.32 GB

Results (1)

#	CreditCardFares
1	1044600010

Copy Download results

Time in queue: 362 ms

Run time: 7.944 sec

Data scanned: 9.32 GB

Select list all the records from the view `cashtrips`:

```
Select * from cashtrips;
```

The screenshot shows the AWS Management Console with the Athena service selected. The query editor interface is open, displaying a completed query. The query text is: `Select * from cashtrips;`. The results section shows one row of data: # 450031761. The results table has a header row labeled '#', 'CashFares'. The bottom of the screen shows standard AWS navigation links like cloudShell, Feedback, and copyright information.

Time in queue: 129 ms

Run time: 8.218 sec

Data scanned: 9.32 GB

Create a new view called `cashtrips` that joins the data:

```
CREATE VIEW cashtrips AS
WITH
    cc AS
        (SELECT sum(fare) AS cctotal,
        vendor
        FROM yellow
        WHERE paytype = '1'
        GROUP BY paytype, vendor),
    cs AS
        (SELECT sum(fare) AS cashtotal,
        vendor, paytype
        FROM yellow
        WHERE paytype = '2'
        GROUP BY paytype, vendor)

    SELECT cc.cctotal, cs.cashtotal
    FROM cc
```

```
JOIN cs
ON cc.vendor = cs.vendor;
```

The screenshot shows the AWS Management Console with the Athena service selected. The top navigation bar includes 'aws', 'Services', 'Search', and 'N. Virginia'. The user is identified as 'vocabs/user2486983=psingenhallprabhu@umassd.edu @ 4284-441...'. The main area is titled 'Amazon Athena > Query editor' and shows the 'Editor' tab selected. A message at the top states 'Athena now supports typeahead code suggestions to speed up SQL query development'. Below this, the query editor displays the following SQL code:

```

Data source: AwsDataCatalog
Database: taxidata
Tables and views: creditcard, jan, yellow
Views: cashtrips, cctrips, comparepay

SQL:
14 GROUP BY paytype, vendor
15
16 SELECT cc.cctotal, cs.cashtotal
17 FROM cc
18 JOIN cs
19 ON cc.vendor = cs.vendor;
20

```

The 'Run again' button is highlighted in orange. The 'Query results' section shows a green status bar indicating 'Completed' with a timestamp of 'Time in queue: 47 ms Run time: 733 ms Data scanned: -'. Below this, a message says 'Query successful.'

Time in queue: 47 ms

Run time: 733 ms

Preview the result from the join:

This screenshot is identical to the one above, showing the same query execution details. The key difference is in the 'Query results' section, which now displays the results of the query:

#	cctotal	cashtotal
1	460097126	199181978
2	584502884	250849783

Below the results table are 'Copy' and 'Download results' buttons.

Task 5: Creating Athena named queries by using CloudFormation

Query:

```
SELECT distance, paytype, fare, tip, tolls, surcharge, total FROM yellow WHERE total >= 100.0 ORDER BY total DESC
```

The screenshot shows the AWS Athena Query Editor interface. At the top, there's a banner message: "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." Below this, the "Editor" tab is selected, showing a single query named "Query 11". The query itself is the one provided in the text above. The results section shows 205,264 rows of data from the "yellow" table, filtered by "total >= 100.0". The columns listed are distance, paytype, fare, tip, tolls, surcharge, and total. The results table has 8 visible rows, with the first few being 861604, 861602, 630462, 625901, 538579, 538481, 404094, and 398474.

Create a new CloudFormation template:

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  AthenaNamedQuery:
    Type: AWS::Athena::NamedQuery
    Properties:
      Database: "taxidata"
      Description: "A query that selects all fares over $100.00 (US)"
      Name: "FaresOver100DollarsUS"
      QueryString: >
        SELECT distance, paytype, fare, tip, tolls, surcharge, total
        FROM yellow
        WHERE total >= 100.0
        ORDER BY total DESC
Outputs:
  AthenaNamedQuery:
    Value: !Ref AthenaNamedQuery
```

```
aws cloudformation validate-template --template-body file://athenaquery.cf.yml
```

```

17:1 YAML Spaces: 2
bash - ip-172-31-33-75.e x Immediate x
voclabs:~/environment $ aws cloudformation validate-template --template-body file://athenaquery.cf.yml
{
  "Parameters": {}
}
voclabs:~/environment $ 

```

Creating the stack:

```
aws cloudformation create-stack --stack-name athenaquery --template-body file://athenaquery.cf.yml
```

```

17:1 YAML Spaces: 2
bash - ip-172-31-33-75.e x Immediate x
voclabs:~/environment $ aws cloudformation validate-template --template-body file://athenaquery.cf.yml
{
  "Parameters": {}
}
voclabs:~/environment $ aws cloudformation create-stack --stack-name athenaquery --template-body file://athenaquery.cf.yml
{
  "StackId": "arn:aws:cloudformation:us-east-1:42844136833:stack/athenaquery/5554cea0-5e70-11ee-be76-0a3397ed6cef"
}
voclabs:~/environment $ 

```

Confirming if CloudFormation stack created:

```
aws athena list-named-queries
```

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  AthenaNamedQuery:
    Type: AWS::Athena::NamedQuery
    Properties:
      Database: "taxidata"
      Description: "A query that selects all fares over $100.00 (US)"
      Name: "FaresOver100DollarsUS"
      QueryString: >
        SELECT distance, paytype, fare, tip, tolls, surcharge, total
        FROM yellow
        WHERE total >= 100.0
        ORDER BY total DESC
    Outputs:
      AthenaNamedQuery:
        Value: !Ref AthenaNamedQuery
17:1 YAML - Spaces: 2
```

```
aws -Tp-172-31-33-75.ec x Immediate x
vclabs:~/environment $ aws cloudformation validate-template --template-body file://athenaquery.cf.yml
{
  "Parameters": []
}
vclabs:~/environment $ aws cloudformation create-stack --stack-name athenaquery --template-body file://athenaquery.cf.yml
{
  "StackId": "arn:aws:cloudformation:us-east-1:428444136833:stack/athenaquery/5554eea0-5e78-11ee-be76-0a3397ed6cef"
}
vclabs:~/environment $ aws athena list-named-queries
{
  "NamedQueryIds": [
    "dc779edd-08b1-49bf-b366-6bc14954c30f",
    "da197945-b93d-4fbe-9727-c46a3a90c72"
  ]
}
vclabs:~/environment $
```

To retrieve the details of the named query:

```
aws athena get-named-query --named-query-id da197945-b93d-4fbe-9f27-c46a3a090cf2
```

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  AthenaNamedQuery:
    Type: AWS::Athena::NamedQuery
    Properties:
      Database: "taxidata"
      Description: "A query that selects all fares over $100.00 (US)"
      Name: "FaresOver100DollarsUS"
      QueryString: >
        SELECT distance, paytype, fare, tip, tolls, surcharge, total
        FROM yellow
        WHERE total >= 100.0
        ORDER BY total DESC
    Outputs:
      AthenaNamedQuery:
        Value: !Ref AthenaNamedQuery
17:1 YAML - Spaces: 2
```

```
bash -Tp-172-31-33-75.ec x Immediate x
vclabs:~/environment $ aws athena list-named-queries
{
  "NamedQueryIds": [
    "dc779edd-08b1-49bf-b366-6bc14954c30f",
    "da197945-b93d-4fbe-9727-c46a3a90c72"
  ]
}
vclabs:~/environment $ aws athena get-named-query --named-query-id dc779edd-08b1-49bf-b366-6bc14954c30f
{
  "NamedQuery": {
    "Name": "FaresOver100DollarsUS",
    "Description": "A query that selects all fares over $100.00 (US)",
    "Database": "taxidata",
    "QueryString": "SELECT distance, paytype, fare, tip, tolls, surcharge, total FROM yellow WHERE total >= 100.0 ORDER BY total DESC\\n",
    "NamedQueryId": "dc779edd-08b1-49bf-b366-6bc14954c30f",
    "NameGroup": "primary"
  }
}
```

Saving the named query ID as a bash variable

The screenshot shows the AWS Cloud9 IDE interface. On the left, the file structure for a CloudFormation stack named 'cloud9-athena' is visible, containing 'athenaquery.yaml' and 'README.md'. The 'athenaquery.yaml' file contains a CloudFormation template defining an Athena named query. The template includes parameters for the database ('taxidata'), query description ('A query that selects all fares over \$100.00 (US)'), and the query itself ('SELECT distance, potype, fare, tip, tolls, surcharge, total FROM yellow WHERE total >= 100.0 ORDER BY total DESC'). The 'Outputs' section defines a named query output. In the bottom terminal window, a bash session is running. It first sets the environment variable \$NQ to the named query ID (dc779edd-08b1-49bf-b366-6bc14954c30f). Then, it uses the AWS CLI command 'aws athena get-named-query --named-query-id \$NQ' to retrieve details of the named query, which are displayed in the terminal.

```
NQ=dc779edd-08b1-49bf-b366-6bc14954c30f
echo $NQ

aws cloudformation validate-template --template-body file:///home/ec2-user/cloud9-athena/athenaquery.yaml --region us-east-1
aws cloudformation create-stack --stack-name cloud9-athena --template-body file:///home/ec2-user/cloud9-athena/athenaquery.yaml --region us-east-1
aws cloudformation describe-stacks --stack-name cloud9-athena --region us-east-1
aws athena get-named-query --named-query-id da197945-b93d-4fbe-9f27-c46a3a098cf2
{
    "NamedQuery": {
        "Name": "Create Database",
        "Database": "default",
        "QueryString": "CREATE DATABASE taxidata",
        "NamedQueryId": "da197945-b93d-4fbe-9f27-c46a3a098cf2",
        "WorkGroup": "primary"
    }
}
vclabs:~/environment $ NQ=dc779edd-08b1-49bf-b366-6bc14954c30f
vclabs:~/environment $ echo $NQ
dc779edd-08b1-49bf-b366-6bc14954c30f
vclabs:~/environment $
```

Conclusion Remarks:

In conclusion, this assignment explores various aspects of AWS Athena and CloudFormation, providing a comprehensive understanding of these vital AWS services.

The first section of the assignment delves into the creation and querying of an AWS Glue database and table in Athena. This foundational step is crucial for data analysis and manipulation, serving as the backbone for subsequent data-related tasks.

Optimization is a key theme in this assignment, and the second section demonstrates the optimization of Athena queries by using buckets. By implementing this strategy, query performance is significantly enhanced, leading to more efficient data processing and reduced query execution times.

Furthermore, the assignment discusses another optimization technique: the use of partitions in Athena queries. Organizing data into partitions not only improves query speed but also contributes to cost reduction, making it an essential concept in the context of AWS Athena.

Athena views are another valuable tool covered in this assignment. This section elaborates on how to create and view data using Athena views. This approach simplifies complex queries and enhances data accessibility, contributing to more efficient data analysis.

The final section of the assignment focuses on the creation of Athena named queries using CloudFormation, with a specific emphasis on utilizing Cloud9. It provides a step-by-step guide, including creating the CloudFormation stack, confirming its creation, retrieving details of the named query, and saving the named query ID as a bash variable.

In summary, this assignment showcases a comprehensive understanding of AWS Athena and CloudFormation, essential tools in the field of data science and tech.