AS201

Astra DB for Cloud Native Applications

Lab Guide

Lab 01: Create an Astra DB Account

1. Connect to https://astra.datastax.com
2. Create a free account (use any email you choose)
3. Create a database called *AS201*
4. Create a keyspace called *class*
5. Choose an available region
6. Copy or download the *secure token* and save it for use later
7. Explore your new database and the Astra UI

END OF LAB

Lab 02: Create a Table

1. Connect to the Astra Dashboard
2. Access your database
3. Open the CQL Console
4. Use the keyspace (class)
5. Create a table called *cars* with the following fields

Field	Type	Key
id	int	primary key
make	text	
model	text	
year	int	

6. Insert some data

```
INSERT INTO cars(id, make, model, year)
  values(1001, 'Dodge', 'Challenger', 1971);
INSERT INTO cars(id, make, model, year)
  values(1002, 'Ford', 'Mustang', 1968);
INSERT INTO cars(id, make, model, year)
  values(1003, 'Chevy', 'Camaro', 1969);
INSERT INTO cars(id, make, model, year)
  values(1004, 'Dodge', 'Daytona', 1969);
INSERT INTO cars(id, make, model, year)
  values(1005, 'Dodge', 'Challenger', 1972);
INSERT INTO cars(id, make, model, year)
  values(1006, 'Ford', 'Mustang', 1971);
INSERT INTO cars(id, make, model, year)
  values(1007, 'Dodge', 'Charger', 1969);
```

7. Run a query and select all cars

END OF LAB

Lab 03: INSERTs, UPSERTs and UPDATEs

1. In this lab make sure your CQL commands only contain the necessary information to complete the task
2. In the *cars* table from the previous lab, use an INSERT to change car, id 1007, from a Dodge Charger to a Dodge Dart
3. Verify that the value has changed
4. Use an UPDATE to change it back
5. Verify that the value has changed back

END OF LAB

LAB 04: Partition Keys

1. Use the *DROP TABLE* command in CQL Shell to delete the cars table
2. Recreate the cars table with:

   a composite partition key: *make & model*
   a clustering column: *id*

3. Populate the table with the same data as before
4. Run a query to retrieve the Camaro form the table
5. Write a query to retrieve all the Dodges from the table

END OF LAB

Lab 05: Clustering Columns

1. Use the table from the previous lab for this exercise because it already has a clustering column
2. Write a query to return all Dodge Challengers and all Ford Mustangs with ids less than 1005
3. Write a query to return all Chevy Camaros
4. Write a query to return all cars in descending id order

END OF LAB

Lab 06: Multiple Clustering Columns

1. Use the *DROP TABLE* command in CQL Shell to delete the cars table
2. Create a new cars table

Field	Type	Key
make	text	partition key
model	text	partition key
miles	int	1st clustering column
year	int	2nd clustering column
color	text	

3. Insert the following data into the table

```
INSERT INTO cars(make, model, miles, year, color)
    values('Ford', 'Mustang', 34000, 1969, 'red');
INSERT INTO cars(make, model, miles, year, color)
    values('Ford', 'Mustang', 40000, 1969, 'green');
INSERT INTO cars(make, model, miles, year, color)
    values('Ford', 'Mustang', 45000, 1968, 'blue');
INSERT INTO cars(make, model, miles, year, color)
    values('Chevy', 'Camaro', 13000, 1969, 'red');
INSERT INTO cars(make, model, miles, year, color)
    values('Chevy', 'Camaro', 31000, 1969, 'yellow');
INSERT INTO cars(make, model, miles, year, color)
    values('Chevy', 'Camaro', 31000, 1971, 'red');
INSERT INTO cars(make, model, miles, year, color)
    values('Chevy', 'Camaro', 60000, 1970, 'blue');
```

4. Write a query to list all Ford Mustangs in order by miles (descending)
5. Write a query to list all Ford Mustangs in order by year (ascending)
6. Write a query to list all Ford Mustangs in order by miles (descending)
7. Write a query to list all Chevy Camaros with more than 15000 miles
8. Write a query to list all Chevy Camaros

END OF LAB

Lab 07: Consistency Level

1. Using the cars table from the previous lab
2. Execute a query to list all cars at Consistency Level ALL
3. Execute a query to list all cars at Consistency Level LOCAL_QUORUM
4. Execute a query to list all cars at Consistency Level THREE
5. Execute a query to list all cars at Consistency Level TWO
6. Execute a query to list all cars at Consistency Level ONE
7. Insert a new car (Chevy ,Camaro, 4000 miles, 2022, yellow) at Consistency Level ALL
8. Insert the same car at Consistency Level LOCAL_QUORUM
9. Insert the same car at Consistency Level THREE
10. Insert the same car at Consistency Level TWO
11. Insert the same car at Consistency Level ONE

END OF LAB

Lab 08: Denormalization

1. Using the data from Lab 06, create a new table that allows the following queries (do not delete the cars table)

   all cars from a specific year

   all cars from 1969 with miles between 25000 and 35000

2. Populate the table

3. Execute the queries

END OF LAB

Lab 9: Storage Attached Indexes

1. Use the cars table from the *Multiple Clustering Columns* lab
2. Create an index that allows querying all cars from a given year
3. Create another index that allows querying all yellow and red cars
4. If you needed to query all *red* cars from *1969* would the best solution be an SAI index or a denormalized table?

END OF LAB

Lab 10: Collections

1. Create table of service offerings (you choose the collection type)

Field	Type	Key
name	text	primary key
price	int	
services	collection	

2. Create INSERT statements and insert the following data

name	price	services
tune up	150	change oil, adjust spark plugs, check tires, change filters
rotate tires	50	rotate tires, check tires
winterize	100	check fluids, check battery, mount snow tires, change oil

3. List all the rows of the table
4. Create a table of cars and their service histories (you choose the collection type)

Field	Type	Key
license	text	primary key
service_history	collection	include month and name of service

5. Create INSERT statements and insert the following data

license	service_history
2FLY	JAN – winterize, MAY – tuneup
CYL8R	JAN – rotate tires, FEB – winterize, MAY – tuneup
HOTROD	May – rotate tires

6. Update the service history for HOTROD to add a tuneup in February
7. List all the rows of the table
8. Explain why you chose the collection types you did

END OF LAB

Lab 11: Collections and SAI

1. Use the *service_offerings* and *cars* tables from the previous lab
2. Use SAI to:
    - Find all service offerings that include *change oil*
    - Find all cars that have been *winterized*
    - Find all cars that have had service in May
    - Find all cars that have had a *tuneup* in May

END OF LAB

Lab 12: User Defined Types (UDTs)

1. Delete the *cars* table
2. Create a *customer* UDT with
    - first
    - last
3. Create a *vehicle* UDT with
    - make
    - model
    - year
4. Create a *cars* table

Field	Type	Key
id	int	primary key
owner	customer	
car	vehicle	

5. Create INSERT statements and insert the following data

id	owner	car
1001	Tony Stark	2010 Audi R8
1002	Bruce Banner	2008 Ford Taurus
1003	Tony Stark	2014 Acura NSX

6. Update Bruce Banners car to a 2012 Jeep Wrangler

END OF LAB

Lab 13: Frozen Types

1. Delete the *cars* table
2. Create a *cars* table using the *customer* and *vehicle* UDTs

Field	Type	Key
owner	customer	primary key
car	vehicle	clustering column

3. Create INSERT statements for the following data

owner	car
Tony Stark	2010 Audi R8
Bruce Banner	2008 Ford Taurus
Tony Stark	2014 Acura NSX

4. Create a query that returns both of Tony Stark's cars
5. Create a DELETE statement that deletes Tony Stark's 2010 Audi R8 from the table

END OF LAB

Lab 14: Tuples

1. Recreate the previous lab using tuples for owner and car instead of UDTs

END OF LAB

Lab 15: Timestamps and Aggregation

1. Create a table to store number of cars and trucks per hour that pass through toll booths

Field	Type	Key
tollbooth	text	primary key
time	timestamp	clustering column
cars	int	
trucks	int	

2. Use the following to insert data:

```
INSERT INTO tolls (tollbooth,time,cars,trucks)
     values('West','2022-07-04 09:00 -0500',890,500);
INSERT INTO tolls (tollbooth,time,cars,trucks)
     values('West','2022-07-04 10:00 -0500',990,480);
INSERT INTO tolls (tollbooth,time,cars,trucks)
     values('Central','2022-07-04 09:00 -0500',600, 590);
INSERT INTO tolls (tollbooth,time,cars,trucks)
     values('Central','2022-07-04 10:00 -0500',520, 490);
INSERT INTO tolls (tollbooth,time,cars,trucks)
     values('East','2022-07-04 09:00 -0500',755,500);
INSERT INTO tolls (tollbooth,time,cars,trucks)
     values('East','2022-07-04 10:00 -0500',600, 310);
```

3. Write a query to return the total number of cars in the Central toll booth
4. Write a query to return the total average number of cars in the Central toll booth
5. Write a query to return the average number of trucks in all three toll booths at 9:00
6. Write a query to return the average number of vehicles (cars and trucks) at the East toll booth

END OF LAB

Lab 16: Counters

1. Create a table that uses counters to store number of cars and trucks that pass through toll booths
2. These are the number of cars and trucks to enter for the toll booths

Toll booth	Type
East	120 cars
East	60 trucks
West	81 cars
East	400 cars
West	100 cars
East	380 trucks
West	73 trucks

3. Write a query to return the number of cars and the number of trucks from the East toll booth
4. Write a query to reduce the number of cars in the West toll booth by 15

END OF LAB

Lab 17: Generate Credentials

1. Use the Astra UI console to generate credentials for the *AS201* database
2. Assign the role *API Read/Write User*
3. Download or copy the generated credentials for later use
4. Download and save the *Secure Bundle*

END OF LAB

Lab 18: Use the Astra DB Data Loader

1. Load the file *worldcities.csv*
2. Set the partition key to `country`
3. Set the first clustering column to `city_ascii`
4. Set the second clustering column to `id`
5. Wait until you get the email notification of a successful data load
6. Run a query to count the number of cities named Washington in the United States

Note: The World Cities Database simplemaps.com is made available under Creative Commons 4.0

END OF LAB

Lab 19: Document API

1. Drop the cars table (if it exists)
2. Open the Swagger UI
3. Create a collection called *cars* in the *class* namespace

   Hint: Request body should be: `{"name":"cars"}`
4. Insert a car document

```
{ "car": {"make":"Chevy","model":"Camaro","year":1969},
  "condition": "good" }
```

5. Insert another car document

```
{"car": {"make":"Ford","model":"Mustang","year":1967,"color":"red"},
 "condition": "fair" }
```

6. Not that the Document Id is returned
7. Use the Swagger UI to retrieve the document you just stored

END OF LAB

Lab 20: GraphQL API

1. Drop the cars table if it exists
2. Connect to the GraphQL Playground
3. Create a new cars table

Field	Type	Key
make	text	partition key
model	text	clustering column
id	int	clustering column
color	text	

4. Add the following data

make	model	id	color
Ford	Mustang	1001	red
Chevy	Camaro	1002	blue
Ford	GTO	1003	orange
Ford	Mustang	1004	red

5. Write a query to find all Fords
6. Write a query to find the color of all Ford Mustangs

END OF LAB

Lab 21: Rest API

1. Drop the cars table if it exists
2. Connect to the SwaggerUI
3. Create a new cars table

Field	Type	Key
make	text	partition key
model	text	clustering column
id	int	clustering column
color	text	

4. Add the following data

make	model	id	color
Ford	Mustang	1001	red
Chevy	Camaro	1002	blue
Ford	GTO	1003	orange
Ford	Mustang	1004	red

5. Write a query to find all Fords
6. Write a query to find the color of all Ford Mustangs

END OF LAB