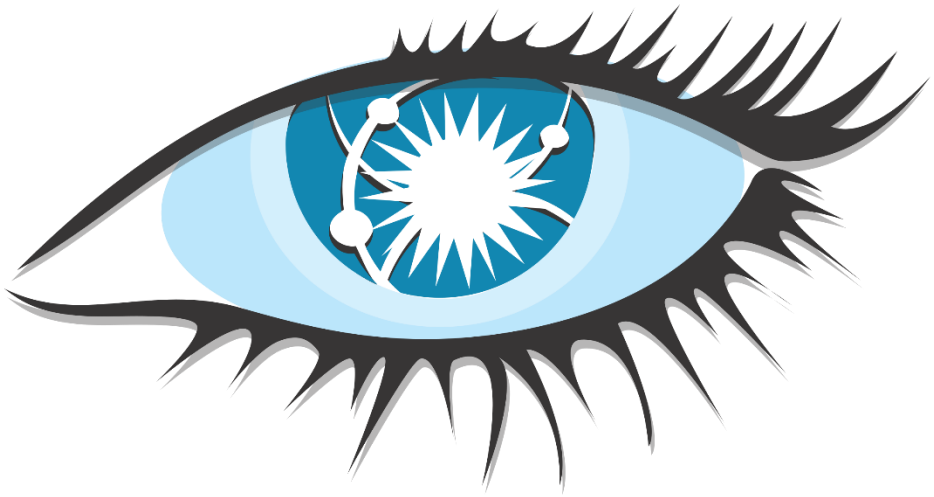
Cassandra



ارائه دهندگان:

رحمت اله انصاری

نیکان میرحسینی

(گروه ۸)

پروژه درس تصمیم‌یار

دکتر ثباتی مقدم

پاییز ۱۴۰۱

فهرست مطالب

فایل‌های مربوط به این پروژه در ریپوزیتوری زیر قرار دارد:

https://github.com/EnAnsari/DSS1401

ایمیل من:

Rahmat2022a@gmail.com

“Ansari”

کاساندرا دیتابیس چیست؟

- آپاچی کاساندرا یک پایگاه داده توزیع شده اوپن سورس NoSQL است که برای مدیریت حجم زیادی از داده‌ها در چندین مرکز داده و فضای ابری ساخته شده است.
- این دیتابیس با زبان Java نوشته شده و توسط Apache توسعه داده شده است.
- Facebook پروژه Cassandra را در جولای ۲۰۰۸، به عنوان یک پروژه متن‌باز بر روی Google code منتشر کرد. در مارس ۲۰۰۹ به یک پروژه Apache incubator و در فوریه ۲۰۱۰ به یک پروژه سطح بالا تبدیل شد. این قابلیت‌های برجسته Cassandra، باعث معروف‌تر شدنش شد.

مهم ترین قابلیت های کاساندرا:

1- ترکیبی

معماری بدون رئیس (masterless) و تأخیر کم به این معنی است که کاسندرا در حالت قطعی کامل ارتباط با مرکز داده، هیچ داده‌ای را از دست نمی‌دهد.

2- تحمل پذیری خطا

پشتیبانی کاساندرا برای تکثیر داده‌ها بین چندین دیتاسنتر، در کلاس خودش بهترین است. کمترین تاخیر برای کاربران را دارد و در نهایت، می‌تواند به شما اطمینان بدهد که قطعی های محلی آسیبی به داده‌هایتان نخواهند رساند. در کاسندرا، می‌توان گره‌های شکست خورده را با گره های سالم جایگزین کرد، بدون اینکه وقفه ای در کار ایجاد شود.

3- توزیع شدگی

Cassandra برای برنامه‌هایی مناسب است که به هیچ عنوان نباید داده‌هایشان را از دست بدهند، حتی زمانی که کل مرکز داده از کار می‌افتد. در کاسندرا هیچ نقطه شکست واحدی وجود ندارد و موقعیت هر گره در خوشه با بقیه برابر است.

4- کارایی بالا

کاسندرا در سنجش معیارها و برنامه‌های واقعی، به دلیل انتخاب های اساسی معماری اش، همیشه، بهتر از جایگزین‌های محبوب NoSQL عمل می‌کند.

5- ذخیره‌سازی انواع داده‌ها:

Cassandra تمام فرمت‌های امکان‌پذیر در داده‌ها را در خود جای داده است، نظیر ساختار یافته، نیمه ساختار یافته و بدون ساختار. در واقع با توجه به نیاز شما، به صورت پویا، ساختار داده را به ساختار مدنظرتان تغییر می‌دهد.

6- سرعت بالا در ذخیره سازی:

Cassandra می‌تواند به طرز چشمگیری، عملیات نوشتن را با سرعت بالایی اجرا کند، همچنین می‌تواند صدها ترابایت داده را بدون کاهش سرعت خواندن، ذخیره کند.

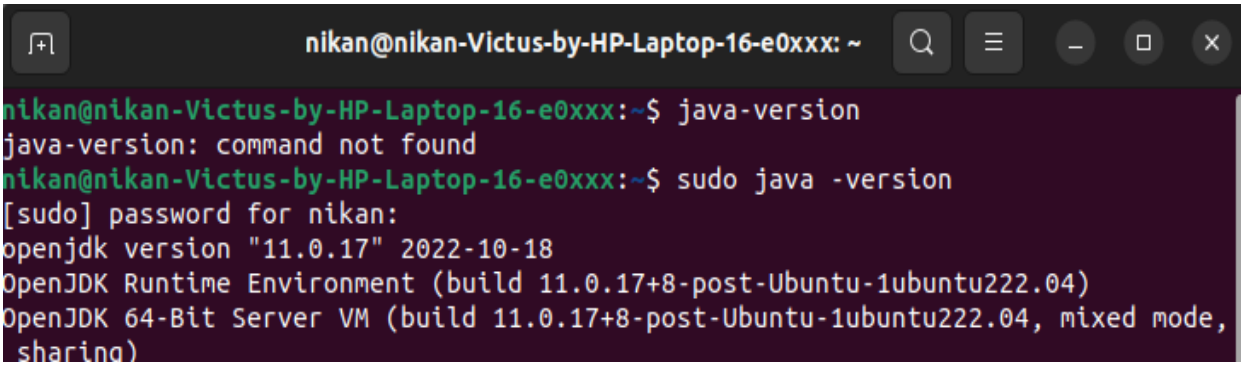
از کاساندرا در نتفلیکس، اینستاگرام، ردیت، اسپاتیفای و اوبر استفاده میشود.

معایب کاساندرا:

1) انتظار می‌رفت در سطوح دسترسی بالاتر، سازگاری بیشتری در این نرم افزار شاهد باشیم.
2) تراکنش‌ها در کاسندرا از ۴ ویژگی مهم ACID پیروی نمی‌کنند.

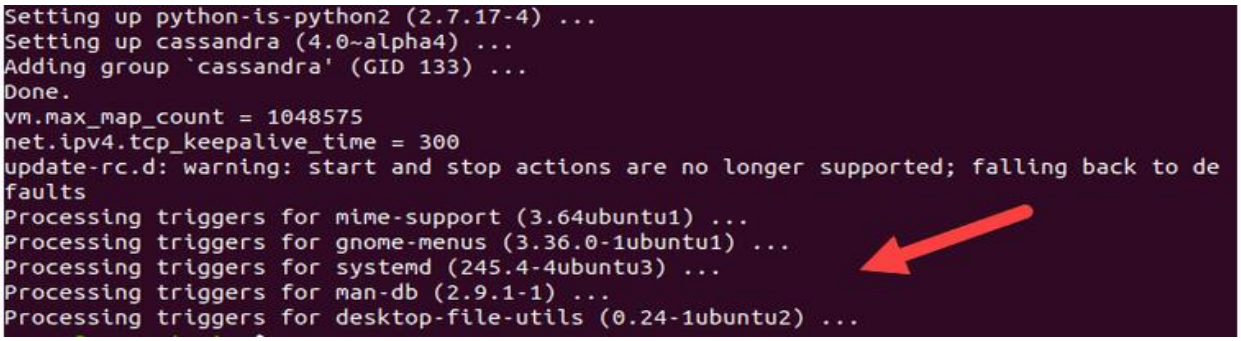
نحوه نصب کاساندرا در اوبونتو:

1. ابتدا برای نصب کاساندرا نیازمند جاوا هستیم، برای نصب جاوا ابتدا باید ریپازیتوری را اپدیت کنیم با دستور زیر: `sudo apt update`
2. سپس با دستور `sudo apt install openjdk-8-jdk -y` جاوا را نصب میکنیم.
3. برای چک کردن نصب بودن جاوا از دستور `sudo java -version` استفاده میکنیم.

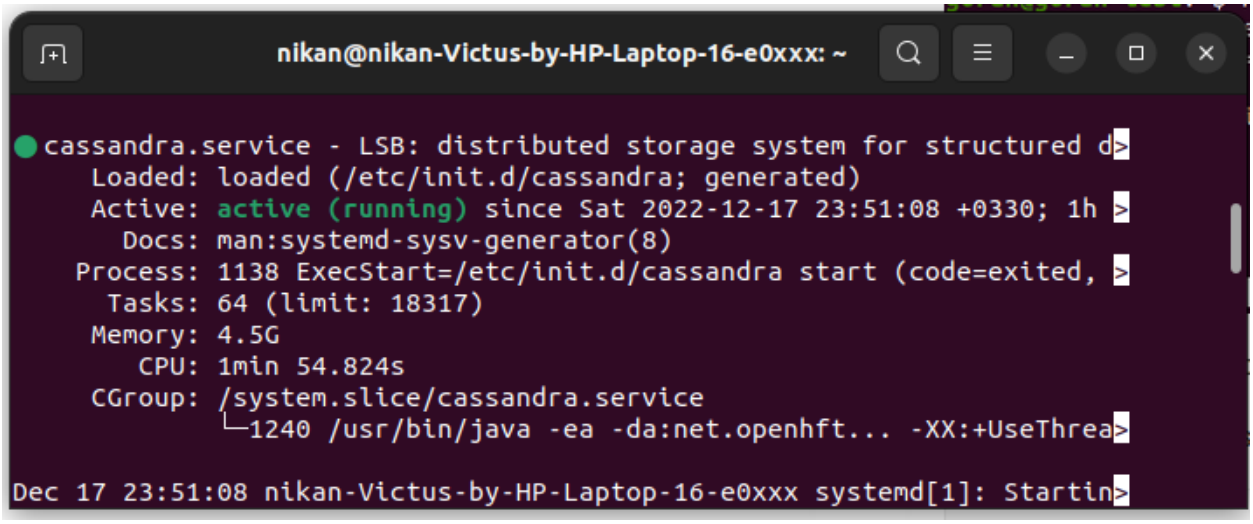


1. حال برای نصب کاسندرا مجدد ریپازیتوری را اپدیت میکنیم و دستور زیر را مینویسیم:

```
sudo apt install Cassandra
```



2. برای چک کردن وضعیت کاساندرا از دستور `sudo systemctl status cassandra` استفاده میکنیم. `Active` به معنای فعال بودن کاساندراست.



برخی دستورات جزئی در کاساندرا:

دستور شروع کاساندرا	`sudo service cassandra start`
دستور پایان کاساندرا	`sudo service cassandra stop`
دستور رفتن به محیط کد نویسی کاساندرا	`cqlsh`

باید برای کار با کاساندرا ابتدا آن را شروع کنیم و با زدن `cqlsh` به محیط آن وارد شویم.

دستورات:

در کاساندرا به `database` اصطلاحا `keyspace` می‌گویند.

برای دیدن `keyspace` های موجود از دستور زیر استفاده می‌کنیم.

```
cqlsh> DESCRIBE KEYSPACES;

system       system_distributed  system_traces  system_virtual_schema
system_auth  system_schema       system_views
```

برای ساخت یک `keyspace` هم از دستور زیر استفاده می‌کنیم.

```
cqlsh> CREATE KEYSPACE test_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}
 AND durable_writes = 'true';
```

در این دستور از اصطلاح `replication` استفاده شده. این به معنی تعداد بک آپ های دیتاهای ماست. اگر مقدار `replicatoin_factor` را چیزی جز یک بگذاریم دیتاهای ما به همان اندازه بک آپ خواهند داشت.

برای استفاده از یک `keyspace` از دستور زیر استفاده می‌کنیم.

```
cqlsh> USE test_keyspace ;
```

برای ساخت جدول هم از دستور زیر استفاده می‌کنیم.

```
cqlsh:test_keyspace> CREATE TABLE employee_by_id (id int PRIMARY KEY , name text , position text ) ;
```

در اینجا پس از عبارات CREATE TABLE نام جدول را وارد می‌کنیم و پس از آن هم نام فیلدهای جدولمان را وارد می‌کنیم. ابتدا نام فیلد و سپس نوع دیتا را وارد می‌کنیم. اگر فیلد مورد نظر کلید اصلی بود عبارت PRIMARY KEY را هم وارد می‌کنیم.

برای دیدن جداول یک keyspace از دستور زیر استفاده می‌کنیم.

```
cqlsh:test_keyspace> DESCRIBE TABLES ;

employee_by_id
```

برای حذف یک جدول هم از دستور زیر استفاده می‌کنیم.

```
cqlsh:test_keyspace> DROP TABLE employee_by_id ;
```

حتی می‌توانیم به صورت زیر هم چند ستون را هم زمان به عنوان PRIMARY KEY تعیین کنیم.

```
cqlsh:test_keyspace> CREATE TABLE employee_by_car_make_sorted (car_make text, age int, id int, name text, PRIM
ARY KEY(car_make, age, id));
```

به صورت زیر هم می‌توان PRIMARY KEY را تعیین کرد.

```
cqlsh:test_keyspace> CREATE TABLE employee_by_car_make_and_car_model (
car_make text, car_model text, id int, name text, PRIMARY KEY ((car_ma
ke, car_model), id));
```

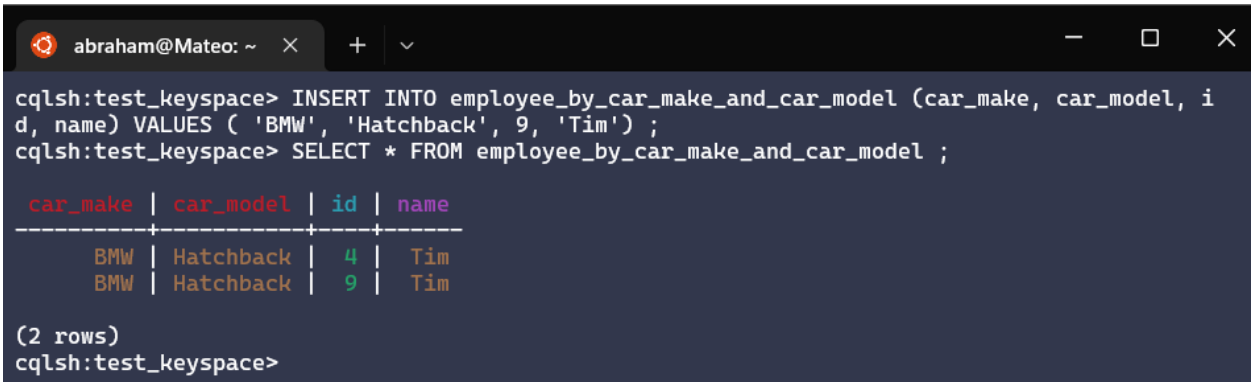
با دستور زیر هم میتوان مشخصات یک جدول را دید:

```
cqlsh:test_keyspace> DESCRIBE TABLE employee_by_id ;

CREATE TABLE test_keyspace.employee_by_id (
    id int PRIMARY KEY,
    name text,
    position text
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

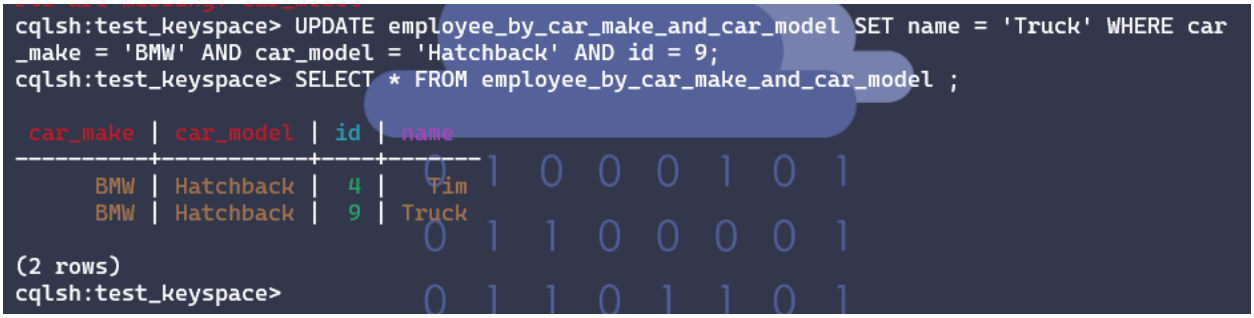
با دستور زیر هم می‌توان یک داده را به جدول درج کرد و اطلاعات جدول را در خروجی چاپ کرد:

```
cqlsh:test_keyspace> SELECT * FROM employee_by_car_make_and_car_model ;

 car_make | car_model | id | name
----------+-----------+----+------

(0 rows)
cqlsh:test_keyspace> INSERT INTO employee_by_car_make_and_car_model (car_make , car_model , id) VALUES ( 'BMW', 'Hatchback', 4);
cqlsh:test_keyspace> SELECT * FROM employee_by_car_make_and_car_model ;

 car_make | car_model | id | name
----------+-----------+----+------
      BMW | Hatchback |  4 | null

(1 rows)
```

در هنگام درج باید تمام مقادیری که به عنوان کلید هستند درج شوند (باقی مقادیر اگر نباشند null میشوند):



مقادیر را می‌توان به صورت پایین هم آپدیت کرد:



در کد زیر هم مقدار یک سطر را به صورت ۱۵ ثانیه‌ای (موقت) تغییر داده ایم:

```
cqlsh:test_keyspace> UPDATE employee_by_car_make USING TTL 15 SET car_model = 'Truck' WHERE
 car_make = 'BMW' AND id=1;
cqlsh:test_keyspace> SELECT * FROM employee_by_car_make;

 car_make | id | car_model
----------+----+-----------
     Benz |  2 |    saloon
      BMW |  1 |    Truck

(2 rows)
cqlsh:test_keyspace> SELECT * FROM employee_by_car_make;

 car_make | id | car_model
----------+----+-----------
     Benz |  2 |    saloon
      BMW |  1 |    Truck

(2 rows)
cqlsh:test_keyspace> SELECT * FROM employee_by_car_make;

 car_make | id | car_model
----------+----+-----------
     Benz |  2 |    saloon
      BMW |  1 |    Truck

(2 rows)
cqlsh:test_keyspace> SELECT * FROM employee_by_car_make;

 car_make | id | car_model
----------+----+-----------
     Benz |  2 |    saloon
      BMW |  1 |     null

(2 rows)
cqlsh:test_keyspace>
```

میتوان در کاساندرا از توابع هم استفاده کرد:

```
cqlsh:test_keyspace> SELECT car_make, id, car_model, writetime(car_model) FROM employee_by_
car_make;

 car_make | id | car_model | writetime(car_model)
----------+----+-----------+----------------------
     Benz |  2 |    saloon |     1671459024419467
      BMW |  1 |     null |                 null

(2 rows)
cqlsh:test_keyspace>
```

می‌توان در یک فیلد چند مقدار هم قرار داد:

```
cqlsh:test_keyspace> UPDATE employee_by_id SET phone = {'343', '565'} WHERE id=1;
cqlsh:test_keyspace> SELECT * FROM employee_by_id;

 id | name | phone           | position
----+------+-----------------+----------
  1 | John | {'343', '565'} |  Manager
  2 |  Bob |            null |      CEO

(2 rows)
```

میتوان از آن مقادیر کم یا اضافه هم کرد:

```
cqlsh:test_keyspace> UPDATE employee_by_id SET phone = phone + {'555'} WHERE id=1;
cqlsh:test_keyspace> SELECT * FROM employee_by_id ;

 id | name | phone                      | position
----+------+----------------------------+----------
  1 | John | {'343', '555', '565'} |  Manager
  2 |  Bob |                       null |      CEO

(2 rows)
```

```
(2 rows)
cqlsh:test_keyspace> UPDATE employee_by_id SET phone = phone - {'555'} WHERE id=1;
cqlsh:test_keyspace> SELECT * FROM employee_by_id ;

 id | name | phone           | position
----+------+-----------------+----------
  1 | John | {'343', '565'} |  Manager
  2 |  Bob |            null |      CEO

(2 rows)
cqlsh:test_keyspace>
```

یا حتی مقادیر آن را خالی کرد:

```
cqlsh:test_keyspace> UPDATE employee_by_id SET phone = {} WHERE id=1;
cqlsh:test_keyspace> SELECT * FROM employee_by_id ;

 id | name | phone | position
----+------+-------+----------
  1 | John |  null |  Manager
  2 |  Bob |  null |      CEO

(2 rows)
cqlsh:test_keyspace>
```

به شکل زیر هم می‌توان یک ستون را حذف کرد:

```
cqlsh:test_keyspace> ALTER TABLE employee_by_id DROP phone ;
cqlsh:test_keyspace> SELECT * FROM employee_by_id ;

 id | name | position
----+------+----------
  1 | John |  Manager
  2 |  Bob |      CEO

(2 rows)
cqlsh:test_keyspace>
```

اگر در هنگام جستجو بخواهیم از فیلدی به غیر از PRIMARY KEY استفاده کنیم می‌بایست به صورت زیر از عبارت زیر استفاده کنیم:

```
cqlsh:test_keyspace> SELECT * FROM employee_by_id WHERE name = 'John' ALLOW FILTERING ;

 id | name | position
----+------+----------
  1 | John |  Manager

(1 rows)
cqlsh:test_keyspace>
```

پیشنهاد ما این است که از این دستور استفاده نشود. به این علت که باید تمام نود در این جستجو چک شود و اینکار هزینه‌بر است.

برای دادن یک آیدی یونیک می‌توان به صورت زیر هم فیلد آیدی را تعریف کرد:

```
cqlsh:test_keyspace> CREATE TABLE employee_by_uuid (id uuid PRIMARY KEY , first_name text,
last_name text);
cqlsh:test_keyspace> SELECT * FROM employee_by_uuid ;

 id | first_name | last_name
----+------------+-----------

(0 rows)
```

مقادیر آن را هم باید به صورت زیر داد:

```
(1 rows)
cqlsh:test_keyspace> INSERT INTO employee_by_uuid (id, first_name , last_name ) VALUES ( uuid(), 'Tim', 'Smith');
cqlsh:test_keyspace> INSERT INTO employee_by_uuid (id, first_name , last_name ) VALUES ( uuid(), 'Bob', 'Hanson');
cqlsh:test_keyspace> SELECT * FROM employee_by_uuid ;

 id                                   | first_name | last_name
--------------------------------------+------------+-----------
 10a4ace3-4f83-4c69-aab9-7c84ae84b9d9 |        Bob |    Hanson
 40a4118b-872e-4517-a9e5-55a0f2ef6e3a |        Tim |     Smith
 02fce97e-5215-4902-952e-afb796486bb7 |        Tom |     Dunne

(3 rows)
cqlsh:test_keyspace> CREATE TABLE employee_by_timeuuid (id timeuuid PRIMARY KEY , first_name text, last_name text);
cqlsh:test_keyspace> SELECT * FROM employee_by_timeuuid;

 id | first_name | last_name
----+------------+-----------

(0 rows)
cqlsh:test_keyspace> INSERT INTO employee_by_timeuuid (id, first_name , last_name ) VALUES
(now(), 'Tim', 'Jones');
cqlsh:test_keyspace> SELECT * FROM employee_by_timeuuid;

 id                                   | first_name | last_name
--------------------------------------+------------+-----------
 8aa75d00-7fac-11ed-83d5-13a4464cd7c9 |        Tim |     Jones

(1 rows)
cqlsh:test_keyspace>
```

همچنین می‌توان مقدار `timeuuid` را هم به آن داد.

```
(1 rows)
cqlsh:test_keyspace> INSERT INTO employee_by_timeuuid (id, first_name , last_name ) VALUES
(now(), 'kate', 'Smith');
cqlsh:test_keyspace> INSERT INTO employee_by_timeuuid (id, first_name , last_name ) VALUES
(now(), 'ally', 'Smith');
cqlsh:test_keyspace> SELECT * FROM employee_by_timeuuid;

 id                                   | first_name | last_name
--------------------------------------+------------+-----------
 e33ad6e0-7fac-11ed-83d5-13a4464cd7c9 |       ally |     Smith
 8aa75d00-7fac-11ed-83d5-13a4464cd7c9 |        Tim |     Jones
 df1ac750-7fac-11ed-83d5-13a4464cd7c9 |       kate |     Smith

(3 rows)
cqlsh:test_keyspace>
```

شبیه هم هستند

متغیر ها را می‌توان به صورت شمارنده هم تعریف کرد:

```
cqlsh:test_keyspace> CREATE TABLE purchaces_by_id (id uuid PRIMARY KEY , purchaces counter)
;
cqlsh:test_keyspace> SELECT * FROM purchaces_by_id ;

 id | purchaces
----+-----------

(0 rows)
cqlsh:test_keyspace> UPDATE purchaces_by_id SET purchaces = purchaces + 1 WHERE id=uuid();
cqlsh:test_keyspace> SELECT * FROM purchaces_by_id ;

 id                                   | purchaces
--------------------------------------+-----------
 b833e66f-269a-4daf-a6e6-a30989d0be1f |         1

(1 rows)
cqlsh:test_keyspace>
```

```
cqlsh:test_keyspace> UPDATE purchaces_by_id SET purchaces = purchaces + 1 WHERE id=uuid();
cqlsh:test_keyspace> SELECT * FROM purchaces_by_id ;

 id                                   | purchaces
--------------------------------------+-----------
 b833e66f-269a-4daf-a6e6-a30989d0be1f |         1

(1 rows)
cqlsh:test_keyspace> UPDATE purchaces_by_id SET purchaces = purchaces + 1 WHERE id=uuid();
cqlsh:test_keyspace> UPDATE purchaces_by_id SET purchaces = purchaces + 1 WHERE id=uuid();
cqlsh:test_keyspace> UPDATE purchaces_by_id SET purchaces = purchaces + 1 WHERE id=uuid();
cqlsh:test_keyspace> SELECT * FROM purchaces_by_id ;

 id                                   | purchaces
--------------------------------------+-----------
 e03e6696-c7a6-4646-a32d-14c21a66a783 |         1
 81273767-f81c-4858-91e3-f8fe63ede5ff |         1
 b833e66f-269a-4daf-a6e6-a30989d0be1f |         1
 534d4287-4537-4b14-a2e8-01d72413b8d3 |         1

(4 rows)
cqlsh:test_keyspace> UPDATE purchaces_by_id SET purchaces = purchaces + 1 WHERE id=534d4287
-4537-4b14-a2e8-01d72413b8d3;
cqlsh:test_keyspace> SELECT * FROM purchaces_by_id ;

 id                                   | purchaces
--------------------------------------+-----------
 e03e6696-c7a6-4646-a32d-14c21a66a783 |         1
 81273767-f81c-4858-91e3-f8fe63ede5ff |         1
 b833e66f-269a-4daf-a6e6-a30989d0be1f |         1
 534d4287-4537-4b14-a2e8-01d72413b8d3 |         2

(4 rows)
cqlsh:test_keyspace>
```

می‌توان تعداد `replication` را هم بالا برد:

```
cqlsh> ALTER KEYSPACE test_keyspace WITH replication = {'class': 'SimpleStrategy', 'replica
tion_factor':'2'} AND durable_writes = 'false';

Warnings :
Your replication factor 2 for keyspace test_keyspace is higher than the number of nodes 1

When increasing replication factor you need to run a full (-full) repair to distribute the
data.

cqlsh> DESCRIBE KEYSPACE test_keyspace ;

CREATE KEYSPACE test_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_f
actor': '2'}  AND durable_writes = false;
```

با استفاده از دستور زیر می‌توان مقادیر یک جدول را خالی کرد:

```
cqlsh> USE test_keyspace ;
cqlsh:test_keyspace> SELECT * FROM employee_by_id;

 id | name | position
----+------+----------
  1 | John |  Manager
  2 |  Bob |      CEO

(2 rows)
```

```
cqlsh:test_keyspace> TRUNCATE employee_by_id;
cqlsh:test_keyspace> SELECT * FROM employee_by_id;

 id | name | position
----+------+----------

(0 rows)
cqlsh:test_keyspace>
```

میتوان مقادیر یک فایل csv را به صورت زیر در یک جدول کاساندرا ریخت:

برای کپی کردن یک جدول در یک csv خروجی هم می‌توان به صورت زیر عمل کرد:

من در این آموزش از wsl2 استفاده کرده‌ام که برای انتقال فایل‌های csv به آن از git و github ام استفاده کرده‌ام.

پیاده کردن یک آنالیز ساده با کاساندرا

پیاده کردن تحلیل این که میانگین درصد دختران نوجوان که از سال ۱۹۹۵ به بعد در تصادفات نقش داشته اند چقدر است:

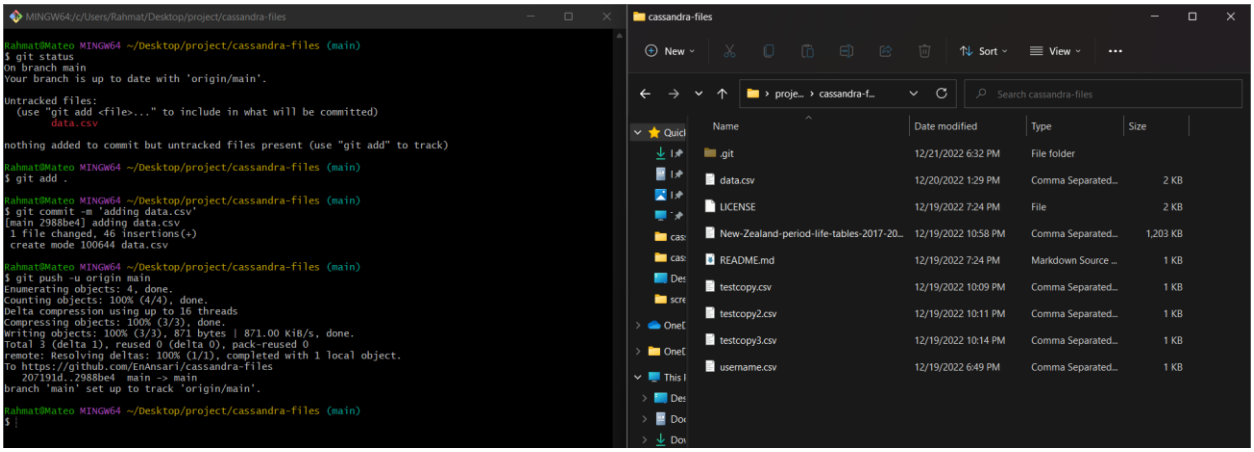
ابتدا دیتای مورد نظر را باید پیدا کنیم. برای ما این دیتا آمار تصادفات نوجوانان (افراد ۱۳ تا ۱۹ ساله) در آمریکا است که بر اساس سال تصادف مرتب شده‌اند.
https://www.iihs.org/topics/fatality-statistics/detail/teenagers#trends
این آمار در سایت بالا موجود است

Motor vehicle crash deaths among 13-19 year-olds by sex, 1975-2020						
Year	Male		Female		Total*	
	Number	%	Number	%	Number	%
1975	6,532	75	2,215	25	8,748	100
1976	6,826	73	2,530	27	9,356	100
1977	6,983	72	2,650	28	9,633	100

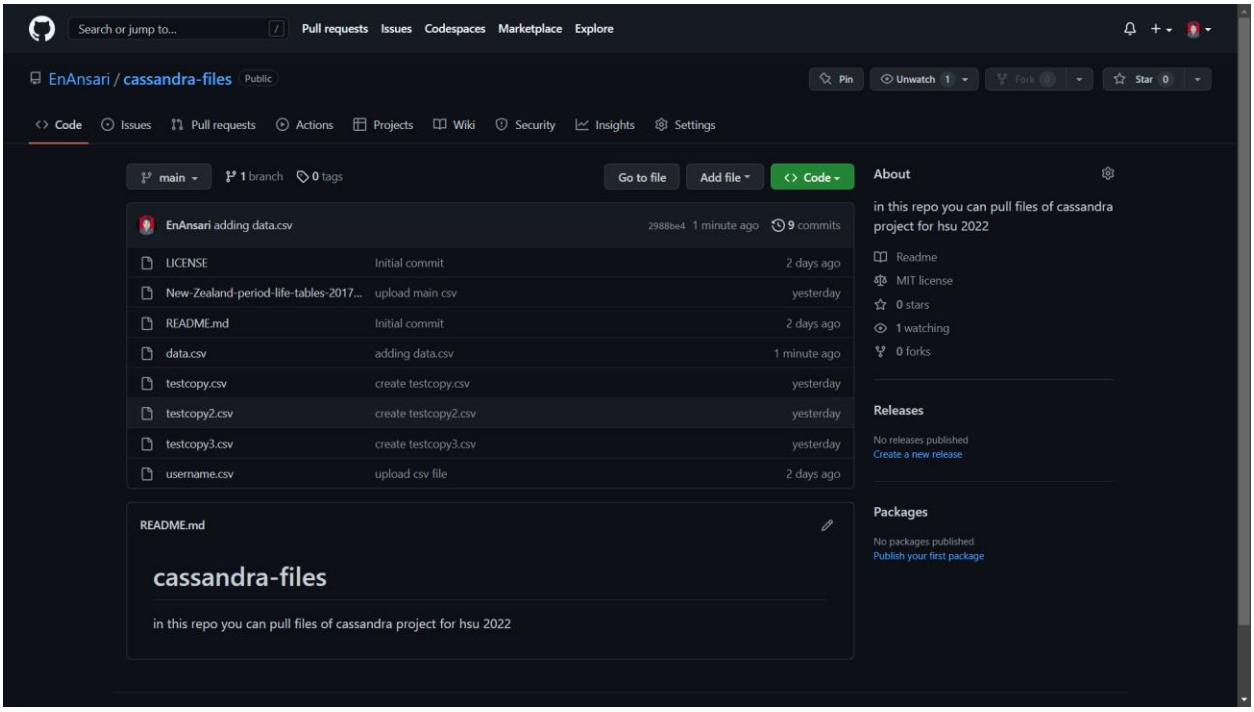
Motor vehicle crash deaths among 13-19 year-olds by sex, 1975-2020						
Year	Male		Female		Total*	
	Number	%	Number	%	Number	%
1978	7,295	73	2,645	27	9,940	100
1979	7,280	73	2,639	27	9,920	100
1980	6,932	73	2,591	27	9,524	100
1981	6,014	72	2,301	28	8,315	100
1982	5,354	73	1,969	27	7,323	100
1983	4,850	71	1,955	29	6,805	100
1984	4,947	71	2,005	29	6,952	100
1985	4,715	70	2,022	30	6,737	100
1986	5,280	71	2,182	29	7,466	100
1987	5,107	70	2,186	30	7,293	100
1988	5,036	70	2,204	30	7,242	100
1989	4,528	68	2,158	32	6,688	100
1990	4,420	69	1,944	31	6,364	100
1991	3,891	68	1,867	32	5,760	100
1992	3,495	67	1,713	33	5,215	100
1993	3,678	68	1,742	32	5,421	100
1994	3,770	67	1,859	33	5,632	100
1995	3,702	65	1,970	35	5,675	100
1996	3,855	66	1,963	34	5,819	100
1997	3,715	65	2,014	35	5,730	100
1998	3,649	65	1,960	35	5,610	100
1999	3,745	65	2,007	35	5,752	100
2000	3,759	66	1,925	34	5,685	100
2001	3,735	67	1,859	33	5,594	100
2002	3,939	66	2,015	34	5,954	100
2003	3,772	66	1,946	34	5,718	100
2004	3,696	65	1,948	35	5,645	100
2005	3,496	66	1,803	34	5,300	100
2006	3,415	66	1,744	34	5,159	100
2007	3,280	66	1,701	34	4,981	100
2008	2,694	66	1,373	34	4,070	100
2009	2,222	64	1,257	36	3,480	100
2010	2,034	65	1,087	35	3,121	100
2011	1,991	66	1,041	34	3,033	100

Motor vehicle crash deaths among 13-19 year-olds by sex, 1975-2020						
Year	Male		Female		Total*	
	Number	%	Number	%	Number	%
2012	1,863	66	972	34	2,837	100
2013	1,661	65	880	35	2,543	100
2014	1,802	69	828	31	2,630	100
2015	1,811	66	935	34	2,747	100
2016	1,867	66	969	34	2,837	100
2017	1,829	66	933	34	2,762	100
2018	1,580	63	914	37	2,496	100
2019	1,589	66	804	34	2,394	100
2020	1,866	68	864	32	2,738	100
*Total includes other and/or unknowns						

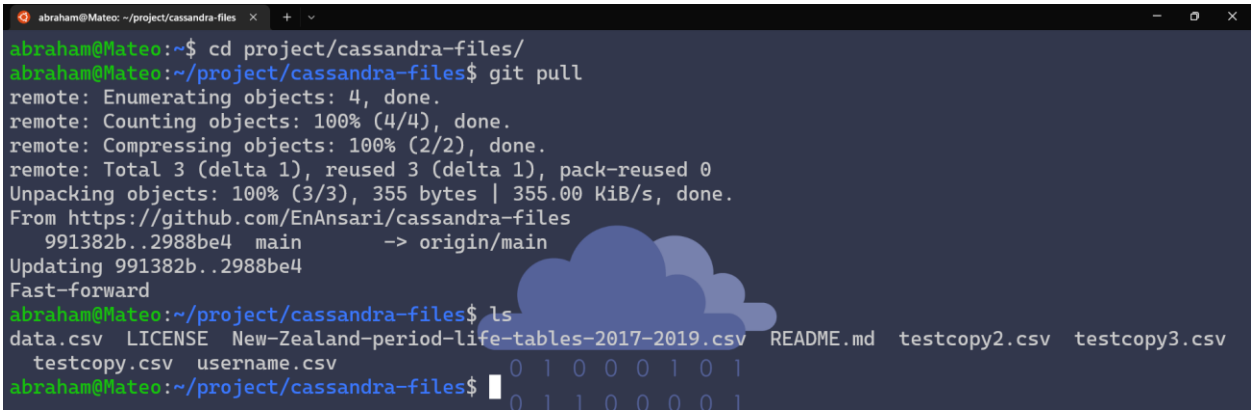
حال این آمار را در `excel` ذخیره و به صورت فایل `csv` ذخیره می‌کنیم. این فایل را هم با استفاده از گیت به گیت هابم منتقل کردم. (نام فایل مورد نظر `data.csv` می‌باشد)



این فایل در گیت هاب پس از دستورات بالا قرار داده شد:

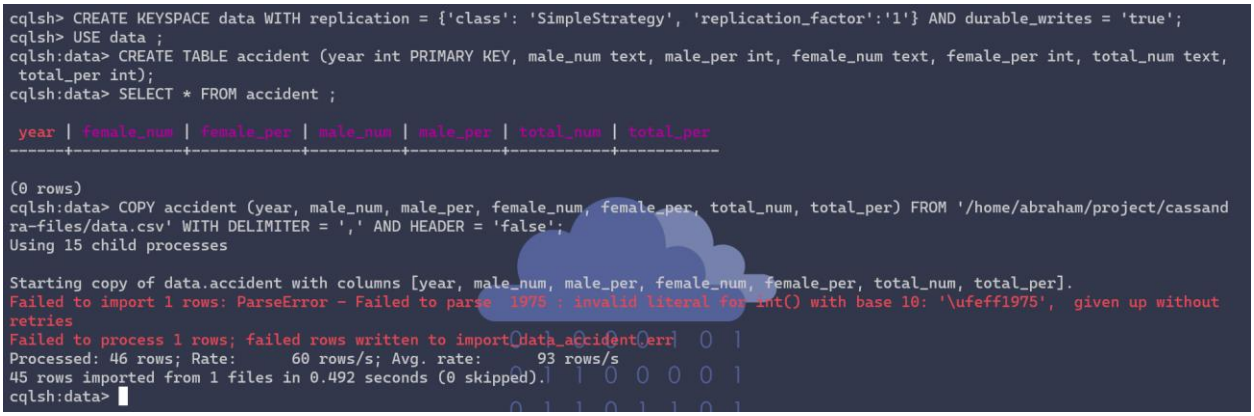


در wsl2 هم این ریپوزیتوری را کلون و سپس در صورت نیاز pull می‌کنیم:



با دستورات زیر کاساندرا را فعال و وارد محیط آن می‌شویم.

با دستورات زیر هم یک keyspace و یک جدول ساختیم و مقادیر فایل csv را در آن کپی کردیم:



در نهایت خروجی مورد نظر را با کد زیر داریم: