Final Project: Milestone 2

CS 327E
April 2, 2018

Announcements:

Demo Day: Friday 04/27
Demo Schedule: https://tinyurl.com/yd68gutt

Friday 04/27 in WAG 420							
Time	Group	Time	Group	Time	Group	Time	Group
8:50:00 AM	EuchDaniel	10:30 AM	Spooky-Data-S-ance	2:00 PM	Juan-Chujun	3:30 PM	CS327E_Alan_Brooks
9:00:00 AM	HarlanHiggins	10:40 AM	ElemOfDatabases	2:10 PM	wk-cs327e-project	3:40 PM	DatabaseWangWhitworth
9:10:00 AM	Jugal-Jacoby	10:50 AM	CKDbSp2018	2:20 PM	cs327e-databases	3:50 PM	ShootingStars_1
9:20:00 AM	KapetanakisSaenzCS327E	11:00 AM	DragonHsieh	2:30 PM	DBYZ327	4:00 PM	candice-and-karla
9:30:00 AM	best_app	11:10 AM	Bayes-the-Databae	2:40 PM	Barney-Bryce-DBProjs	4:10 PM	AVID
9:40:00 AM	Databases-Spring-2018	11:20 AM	Jean-Remy_Nikhil	2:50 PM	LBData	4:20 PM	ArenaOfValor
9:50:00 AM	Tieu-Nguyen	11:30 AM	TeamAK	3:00 PM	AnuragMax-repo	4:30 PM	TranWangDDS
10:00:00 AM	MJdatabases	11:40 AM	JJ	3:10 PM	CS327E-Wu-	4:40 PM	DBVM
10:10:00 AM	DBAssignments	11:50 AM	ChzGrater	3:20 PM	brogrammers	4:50 PM	hyeongju_alex
10:20:00 AM	BigH					5:00 PM	carr-clark-cs327e

1) What is a key characteristic of Dremel's data model?

A) It is a flat relational model
B) It is a graph-like data model
C) It is a nested record data model

2) Does Dremel enforce a schema on the records of a table?

A) Yes, records must satisfy a schema that applies to the table
B) No, records can have different structures in the same table

3) Dremel stores records in a _____ format.

A) row format
B) column format
C) JSON format

4) What is a key performance benefit of column striping?

A) columns that are not needed by the query are not read from disk.
B) rows that are not needed by the query are not read from disk.

5) What is Dremel’s query language?

A) Standard SQL + path expressions for querying nested fields
B) MapReduce
C) Cypher
D) None of the above

BigQuery & UNION ALL

```
SELECT c1, c2, c3
FROM `project.dataset.T1`
[WHERE c1 > c2]
UNION ALL
SELECT c1, c2, c3
FROM `project.dataset.T2`
[WHERE c2 > c3]
[ORDER BY c1];
```

Note: UNION ALL is not available in legacy SQL and requires turning on the Standard SQL option in BigQuery.

austin.Bookings

listing_id	date	price
1078	2017-03-01	300
1078	2017-03-02	305
1078	2017-03-12	700
2265	2017-08-07	250

portland.Bookings

listing_id	date	price
7893	2017-11-05	100
7893	2017-07-23	200
7893	2017-07-06	250
9356	2017-05-21	60

```
SELECT listing_id, date, price
FROM `utcs-spr2018.austin.Bookings`
UNION ALL
SELECT listing_id, date, price
FROM `utcs-spr2018.portland.Bookings`;
```

Result Table

listing_id	date	price
1078	2017-03-01	300
1078	2017-03-02	305
1078	2017-03-12	700
2265	2017-08-07	250
7893	2017-11-05	100
7893	2017-07-23	200
7893	2017-07-06	250
9356	2017-05-21	60

austin.Bookings

listing_id	date	price
1078	2017-03-01	300
1078	2017-03-02	305
1078	2017-03-12	700
2265	2017-08-07	250

portland.Bookings

listing_id	date	price
7893	2017-11-05	100
7893	2017-07-23	200
7893	2017-07-06	250
9356	2017-05-21	60

```
SELECT 'Austin' as city, listing_id, date, price
FROM `utcs-spr2018.austin.Bookings`
UNION ALL
SELECT 'Portland' as city, listing_id, date, price
FROM `utcs-spr2018.portland.Bookings`;
```

Result Table

city	listing_id	date	price
Austin	1078	2017-03-01	300
Austin	1078	2017-03-02	305
Austin	1078	2017-03-12	700
Austin	2265	2017-08-07	250
Portland	7893	2017-11-05	100
Portland	7893	2017-07-23	200
Portland	7893	2017-07-06	250
Portland	9356	2017-05-21	60

BigQuery & SQL Functions

Types of functions:

- Datatype conversion functions
- String functions
- Date functions
- Time functions
- Aggregate functions
- Approximate Aggregate functions
- Statistical functions
- Mathematical functions
- Array functions

Examples:

```
CAST()
CONCAT()
LENGTH()
ENDS_WITH()
SUBSTR()
CURRENT_DATE()
DATE_DIFF()
ROUND()
RAND()
STDDEV()
APPROX_QUANTILES()
```

Reference Documentation: https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators

New Query

Query Editor | UDF Editor

```sql
select concat(cast(latitude as string), ",", cast(longitude as string)) as location, avg(price) as avg_price
from `utcs-spr2018.austin.Listing`
where price > 0
group by location
order by avg_price desc;
```

SQL

Ctrl + Enter: run query, Tab or Ctrl + Space: autocomplete.

Standard SQL Dialect

RUN QUERY | Save Query | Save View | Format Query | Show Options

Query complete (2.0s elapsed, 313 KB processed)

Results | Details

Download as CSV | Download as JSON | Save as Table | Save to Google Sheets

Row	location	avg_price
1	30.344606868620414,-97.813255098913857	10000.0
2	30.295701597087589,-97.753720327408686	10000.0
3	30.181429832584236,-97.815041473852219	10000.0
4	30.272496456764831,-97.773868684092463	9999.0
5	30.346512962861457,-97.87715815914	9594.0

Table | JSON

First < Prev Rows 1 - 5 of 13367 Next > Last

Tip: Create views in BigQuery by running the query and choosing the Save View option.

Subqueries: General Form

```
SELECT c1, c2, c3
FROM (Table Subquery)
WHERE c1 >= c2;

SELECT c1, c2, c3
FROM T1
WHERE c1 IN (Column Subquery);

SELECT c1, c2, (Scalar Subquery)
FROM T1
WHERE c1 = (Scalar Subquery);
```

Note: Subqueries require turning on the *Standard SQL* option in BigQuery.

New Query

```
1  select l.id, l.name, l.street, l.price, l.host_id
2  from `utcs-spr2018.austin.Listing` l
3  where l.id IN
4    (select wa.listing_id from
5       (select listing_id
6        from `utcs-spr2018.austin.Amenity`
7        where amenity_name = 'Washer') as wa
8     join
9       (select listing_id
10       from `utcs-spr2018.austin.Amenity`
11       where amenity_name = 'Dryer') as da
12    on wa.listing_id = da.listing_id)
13 order by l.price;
```

Note: A subquery is required for finding the listings that have **both** Washer and Dryer amenities.

Standard SQL Dialect

RUN QUERY | Save Query | Save View | Format Query | Show Options | Query complete (2.7s elapsed, 4.05 MB processed)

Results | Details | Download as CSV | Download as JSON

Row	id	name	street	price	host_id
1	16845383	Comfy Room in Trendy South Austin	Run of the Oaks Street	10.0	15494878
2	9214127	Luxurious twin bunk bed (Top Bunk)	McNeil Drive	14.0	47037305
3	15671568	Cozy and inviting space in Northwest Austin	Meadgreen Circle	14.0	101207351
4	15034794	Amazing deal in a comfortable apartment	North Lamar Boulevard	15.0	8167447

New Query

Query Edit

```sql
select id, name, since, response_time, response_rate,
 (select max(price) from `utcs-spr2018.austin.Listing` l
  join `utcs-spr2018.austin.Host` hh on l.host_id = hh.id
  where hh.id = h.id) as max_price
from `utcs-spr2018.austin.Host` h
where identity_verified = True and is_superhost = True
and response_rate = 100 and state = 'Texas' and city = 'Austin'
order by max_price;
```

Note: A subquery is required for computing the max price.

Standard SQL Dialect

Ctrl + Enter: run query, Tab or Ct

RUN QUERY | Save Query | Save View | Format Query | Show Options

Query complete (2.4s elapsed, 781 KB processed)

Results | Details | Download as CSV | Download as JSON | Save as Table | Sav

Row	id	name	since	response_time	response_rate	max_price
1	9108103	Mel And Nicole	2013-09-28	within an hour	100	22.0
2	59452947	Gabriel	2016-02-19	within a few hours	100	30.0
3	56533216	Barbra	2016-01-28	within an hour	100	34.0
4	6858592	Amanda	2013-06-11	within an hour	100	34.0