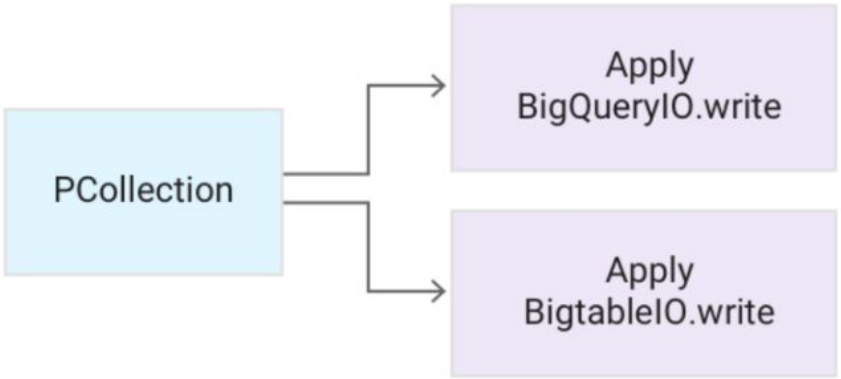
CS 327E Class 10

November 26, 2018

Announcements

- Scheduling your group presentation for Milestone 10. All presentations will happen on week of 12/10 M-F in the evenings. Send me your preferred days/times by **Friday**.
- How to get feedback on your cross-dataset queries and pipeline designs today. Sign-up sheet: https://tinyurl.com/y9fdoggk

1) What is meant by the following usage pattern?



A. The elements in the PCollection are split up such that 1/2 elements are written to BigQuery and 1/2 are written to Bigtable.
B. The same PCollection can be written to multiple data sinks including BigQuery and Bigtable.
C. The PCollection can only be written to BigQuery or Bigtable.

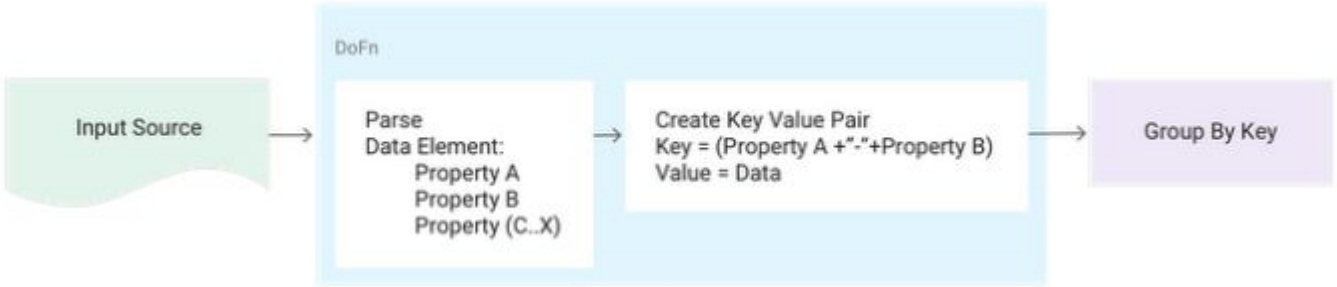
2) How do the authors suggest handling bad data?

A. Send the bad data out of the DoFn as a SideOutput in a try-catch block.
B. Send the bad data into the DoFn as a SideInput.
C. Log the bad data without writing it to a back-end database.

3) What method do the authors suggest for triggering a Dataflow pipeline that needs to start after a file has been uploaded to Google Cloud Storage?

A. Use a simple REST endpoint to trigger the pipeline.
B. Open CloudShell and run the pipeline from the command-line.
C. Trigger the pipeline from Google Cloud Storage.

4) What is meant by the following usage pattern?



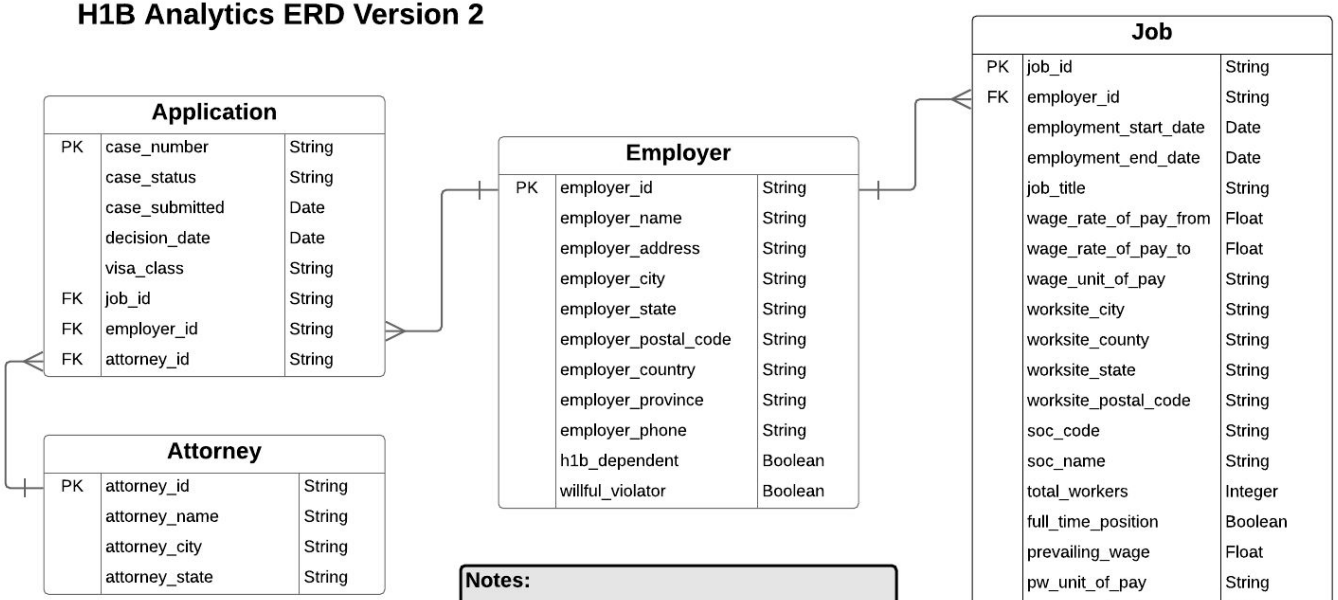
A. GroupByKey requires a preceding DoFn step in the pipeline.
B. GroupByKey requires a composite key as input.
C. Create a composite key to group by multiple properties using GroupByKey.

5) What method do the authors suggest for joining two PCollections in which one of the PCollections is small?

A. Use a CoGroupByKey transform
B. Use a SideInput to a ParDo
C. Use a SQL Join

Case Study: Part 2

H1B Analytics ERD Version 2



Notes:

Source Tables: h1b_split.Employer_Temp, h1b_split.Application_Temp, h1b_split.Job_Temp

Target Tables: h1b_split.Employer, h1b_split.Application, h1b_split.Job. All new tables created and populated from Beam pipelines.

Changes since previous version:
- Removed **187,117** duplicate records from Employer table based on uniqueness criteria of (employer name, city) pairs.
- Added reference to employer_id from Job and Application tables.

Number of Rows		
	v1	**v2**
Employer	348,876	161,759
Job	2,230,779	2,230,625
Application	2,633,426	2,633,156
Attorney	19,861	N/A

Second Dataset

Table Details: Corporate_Registrations_CA

Schema | Details | Preview

so_file_number	STRING
corporation_number	INTEGER
corporation_status	STRING
corporation_classification	STRING
corporation_name	STRING
care_of_name	STRING
mail_address_line_1	STRING
mail_address_line_2	STRING
mail_address_city	STRING
mail_address_state_or_country	STRING
mail_address_zip_code	STRING
corporation_type	STRING
incorporation_date	DATE
so_file_date	DATE
term_expiration_date	DATE
chief_executive_officer_name	STRING

chief_executive_officer_address_line_1	STRING
chief_executive_officer_address_line_2	STRING
chief_executive_officer_address_city	STRING
chief_executive_officer_address_state_or_county	STRING
chief_executive_officer_address_zip_code	STRING
agent_name	STRING
agent_address_line_1	STRING
agent_address_line_2	STRING
agent_address_city	STRING
agent_address_state_or_county	STRING
agent_address_zip_code	STRING
state_or_foreign_country	STRING
ftb_suspension_status	STRING
corporation_tax_base	STRING
transaction_julian_date	DATE
ftb_suspension_string	STRING
filler	STRING

State Table Details:

AZ: 225 MB size, 869,943 rows
CA: 1.1 GB size, 3,792,457 rows
CO: 38 MB size, 160,808 rows
CT: 192 MB size, 796,877 rows
GA: 302 MB size, 2,076,016 rows;
116 MB size, 2,063,919 rows
MA: 221 MB size, 1,066,639 rows
MN: 374 MB size, 1,688,714 rows;
799 MB size, 4,072,355 rows
MO: 133 MB size, 2,364,476 rows;
519 MB size, 2,115,151 rows
NC: 262 MB size, 1,389,877 rows
OH: 497 MB size, 2,408,556 rows
NY: 512 MB size, 2,587,015 rows
VA: 111 MB size, 334,008 rows
WA: 205 MB size, 1,152,309 rows

Table Schemas:

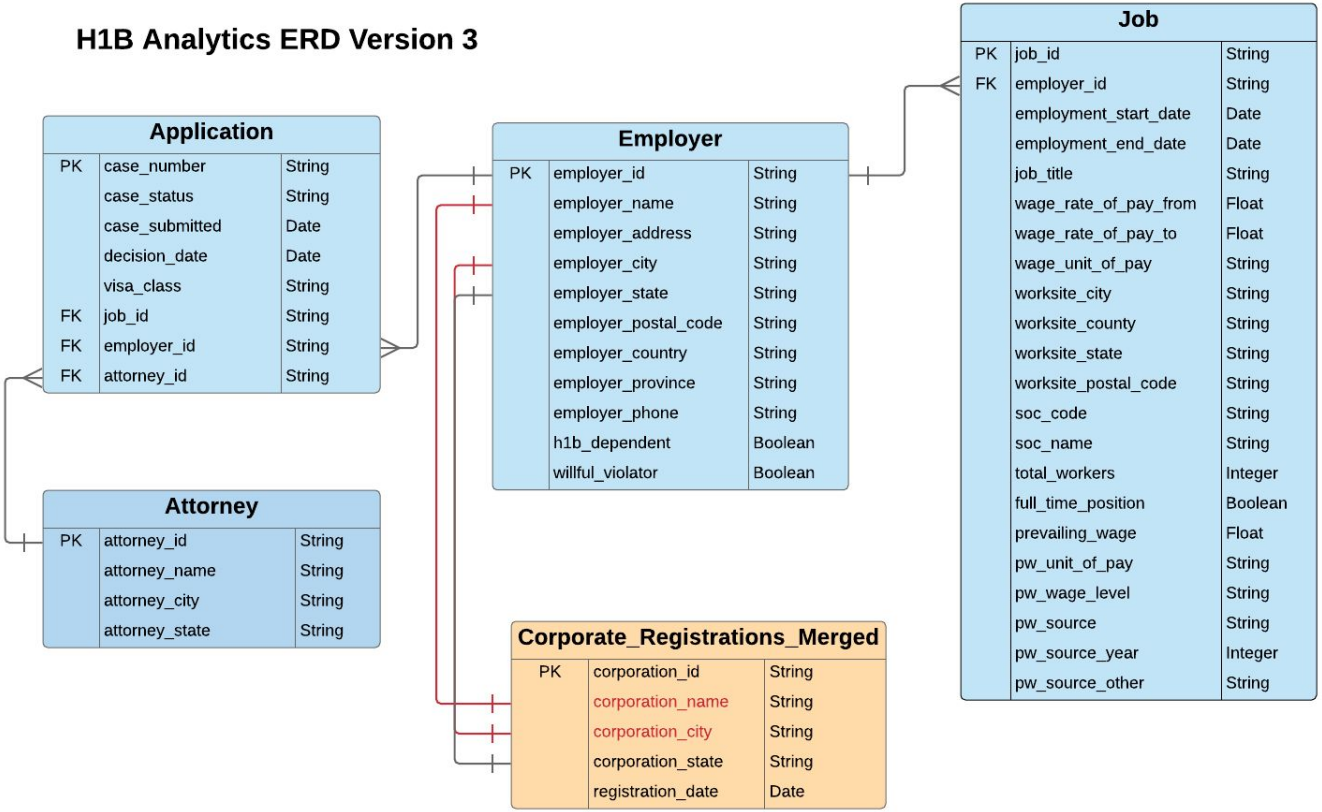
-Each state has unique schema for tracking its corporate registrations.
-Consistent schema for subset of fields successfully derived through CTAS.

SQL Transforms

```sql
create table sec_of_state.Corporate_Registrations_Merged
(
    corporation_id STRING,
    corporation_name STRING,
    corporation_city STRING,
    corporation_state STRING,
    registration_date DATE,
    empty_date DATE
)
PARTITION BY empty_date
CLUSTER BY corporation_state;

create table sec_of_state.Corporate_Registrations_Cleaned
(
    corporation_id STRING,
    corporation_name STRING,
    corporation_city STRING,
    corporation_state STRING,
    registration_date DATE,
    empty_date DATE
)
PARTITION BY empty_date
CLUSTER BY corporation_state;

--AZ
insert into sec_of_state.Corporate_Registrations_Merged (corporation_id, corporation_name, corporation_city, corporation_state, registration_date)
select distinct File_Number, Corporation_Name, First_Address_City, 'AZ', Date_of_Incorporation
from sec_of_state.Corporate_Registrations_AZ
where First_Address_State = 'AZ'
order by corporation_name;

--CA
insert into sec_of_state.Corporate_Registrations_Merged (corporation_id, corporation_name, corporation_city, corporation_state, registration_date)
select CAST(corporation_number as STRING), corporation_name, mail_address_city, 'CA', incorporation_date
from sec_of_state.Corporate_Registrations_CA
where corporation_type = 'Articles of Incorporation'
and mail_address_state_or_country = 'CA'
order by corporation_name;
```

Source File: https://github.com/shirleycohen/h1b_analytics/blob/master/corporate_registrations_ctas.sql

H1B Analytics ERD Version 3



Notes:

New Source Tables:
sec_of_state.Corporate_Registrations_<state>
where <state> = AZ, CA, CO, CT, GA, MA, MN, MO, NC, NY, OH, VA, WA.
Each state table was loaded from a CSV file.
Most of the states had one table, a few had two.

New Target Table:
-sec_of_state.Corporate_Registrations_Merged
-created and populated from CTAS statements.
-390 MB in size with 16,379,107 rows.

Issues with Target Table:
- corporation_name and corporation_city contain punctuation marks; corporation_name contains suffixes (LLC, INC, etc.)
- only **804** results returned from joining Corporate_Registrations_Merged and Employer on name and city.

Beam Transforms

```
PROJECT_ID = os.environ['PROJECT_ID']

# Project ID is needed for BigQuery data source, even for local execution.
options = {
    'project': PROJECT_ID
}
opts = beam.pipeline.PipelineOptions(flags=[], **options)

with beam.Pipeline('DirectRunner', options=opts) as p:

    query_str = 'SELECT corporation_id, corporation_name, corporation_city, corporation_state, registration_date ' \
                'FROM `sec_of_state.Corporate_Registrations_Merged` LIMIT 100'

    query_results = p | 'Read from BQ CorpReg' >> beam.io.Read(beam.io.BigQuerySource(query=query_str, use_standard_sql=True))

    query_results | 'Write to File 1' >> WriteToText('output_query_results.txt')

    clean_pcoll = query_results | 'Transform CorpReg Record' >> beam.ParDo(TransformCorpRegRecord())

    clean_pcoll | 'Write to File 2' >> WriteToText('output_bq_records.txt')

    qualified_table_name = PROJECT_ID + ':sec_of_state.Corporate_Registrations_Cleaned'
    table_schema = 'corporation_id:STRING,corporation_name:STRING,corporation_city:STRING,corporation_state:STRING,registration_date:DATE'

    clean_pcoll | 'Write to BQ CorpReg' >> beam.io.Write(beam.io.BigQuerySink(qualified_table_name,
                                                schema=table_schema,
                                                create_disposition=beam.io.BigQueryDisposition.CREATE_NEVER,
                                                write_disposition=beam.io.BigQueryDisposition.WRITE_TRUNCATE))
```

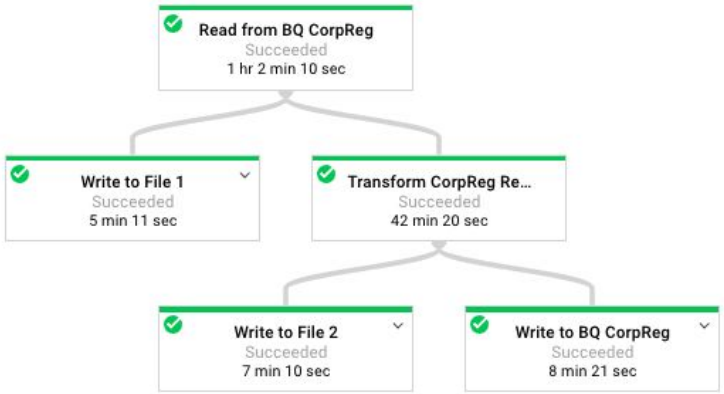
Source File: https://github.com/shirleycohen/h1b_analytics/blob/master/transform_corpreg_table_single.py

Beam Transforms

```
options = {
    'runner': 'DataflowRunner',
    'job_name': 'transform-corp-reg-table',
    'project': PROJECT_ID,
    'temp_location': BUCKET + '/temp',
    'staging_location': BUCKET + '/staging',
    'machine_type': 'n1-standard-8',
    'num_workers': 12
}
opts = beam.pipeline.PipelineOptions(flags=[], **options)

with beam.Pipeline('DataflowRunner', options=opts) as p:

    query_str = 'SELECT corporation_id, corporation_name, corporation_city, corporation_state, registration_date ' \
                'FROM `sec_of_state.Corporate_Registrations_Merged` WHERE corporation_name IS NOT NULL AND corporation_city IS NOT NULL'

    query_results = p | 'Read from BQ CorpReg' >> beam.io.Read(beam.io.BigQuerySource(query=query_str, use_standard_sql=True))

    query_results | 'Write to File 1' >> WriteToText(DIR_PATH + 'output_query_results.txt')

    clean_pcoll = query_results | 'Transform CorpReg Record' >> beam.ParDo(TransformCorpRegRecord())

    clean_pcoll | 'Write to File 2' >> WriteToText(DIR_PATH + 'output_bq_records.txt')

    qualified_table_name = PROJECT_ID + ':sec_of_state.Corporate_Registrations_Cleaned'
    table_schema = 'corporation_id:STRING,corporation_name:STRING,corporation_city:STRING,corporation_state:STRING,registration_date:DATE'

    clean_pcoll | 'Write to BQ CorpReg' >> beam.io.Write(beam.io.BigQuerySink(qualified_table_name,
                                                schema=table_schema,
                                                create_disposition=beam.io.BigQueryDisposition.CREATE_NEVER,
                                                write_disposition=beam.io.BigQueryDisposition.WRITE_TRUNCATE))
```

Source File: https://github.com/shirleycohen/h1b_analytics/blob/master/transform_corpreg_table_cluster.py

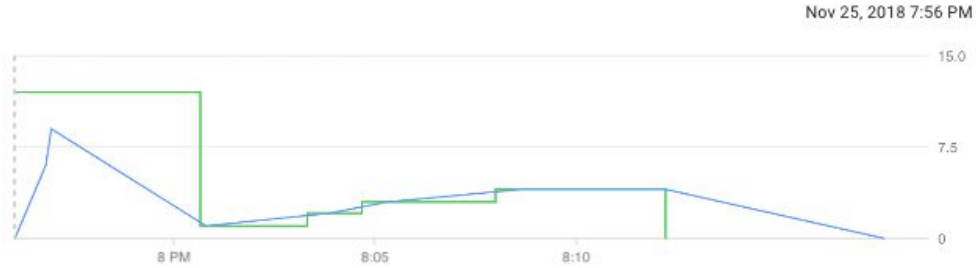
Dataflow Execution



Job summary

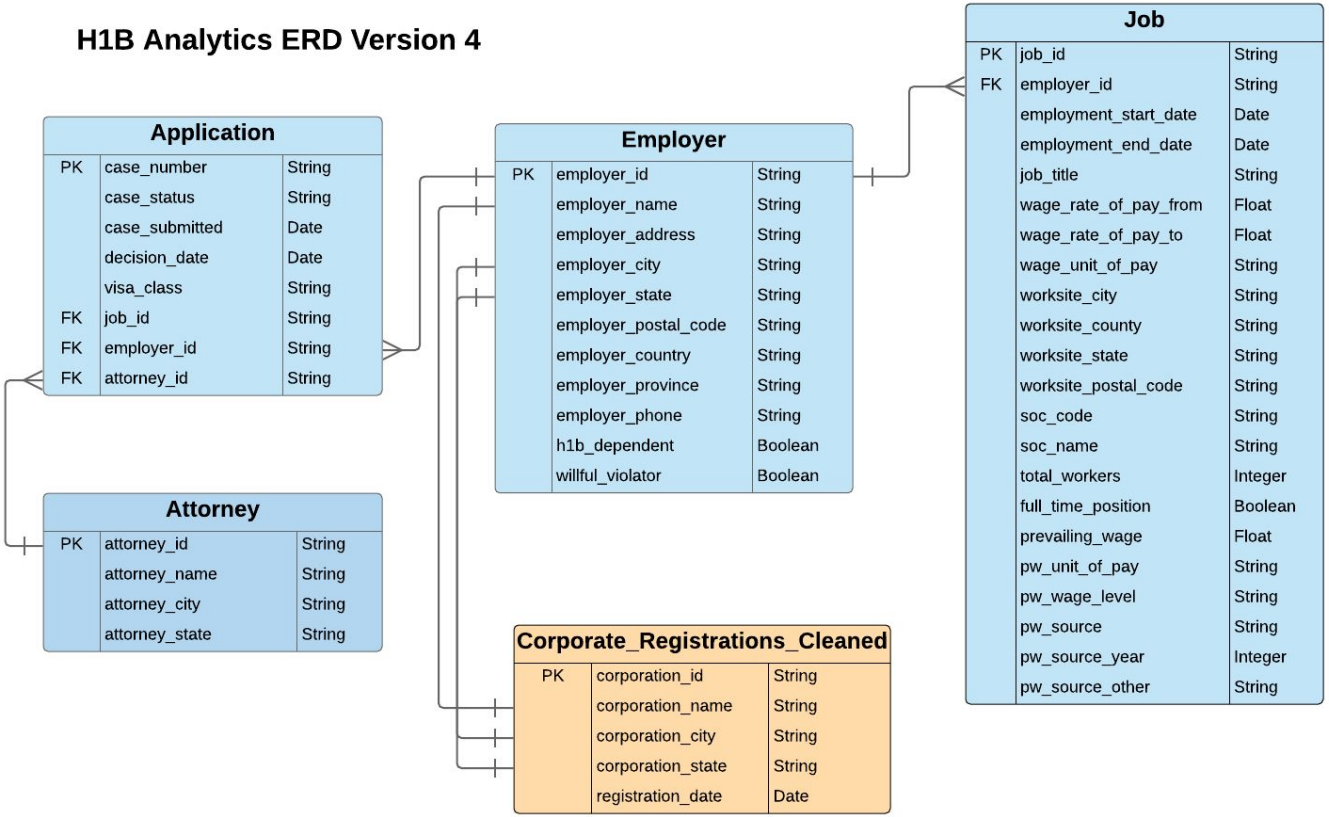
Job name	transform-corp-reg-table
Job ID	2018-11-25_17_55_55-2850952765719790096
Region	us-central1
Job status	Succeeded
SDK version	Google Cloud Dataflow SDK for Python 2.5.0
Job type	Batch
Start time	Nov 25, 2018, 7:55:57 PM
Elapsed time	16 min 20 sec

Worker history



Target workers	Timestamp	Rationale
0	Nov 25, 2018, 8:08:08 PM	Stopping worker pool.
4	Nov 25, 2018, 8:04:58 PM	Autoscaling: Raised the number of workers to 4 based on the rate of progress in the currently running step(s).
3	Nov 25, 2018, 8:02:28 PM	Autoscaling: Raised the number of workers to 3 based on the rate of progress in the currently running step(s).
2	Nov 25, 2018, 8:01:28 PM	Autoscaling: Raised the number of workers to 2 based on the rate of progress in the currently running step(s).
1	Nov 25, 2018, 7:59:28 PM	Autoscaling: Reduced the number of workers to 1 based on the rate of progress in the currently running step(s).
12	Nov 25, 2018, 7:56:02 PM	Starting a pool of 12 workers.

H1B Analytics ERD Version 4



Notes:

New Source Tables:
sec_of_state.Corporate_Registrations_Merged.

New Target Table:
-sec_of_state.Corporate_Registrations_Cleaned.
-generated from Beam pipeline.

Changes since previous version:
- removed punctuation marks and suffixes from corporation_name.
- performed simple validation of corporation_city.
- cross-dataset join returns **12,856** results (instead of only 804 results).

Number of Rows		
	v1	**v2**
Corporate_Registrations	16,379,107	16,321,932
Employer	348,876	161,759
v_Tech_Employer_13_States	29,658	

Cross-Dataset Queries

v_Tech_Employer_Age:

- Joins Employer and Corporate Registrations on name and state
- Calculates age of employer from registration_date

v_Tech_Employer_Age_Label:

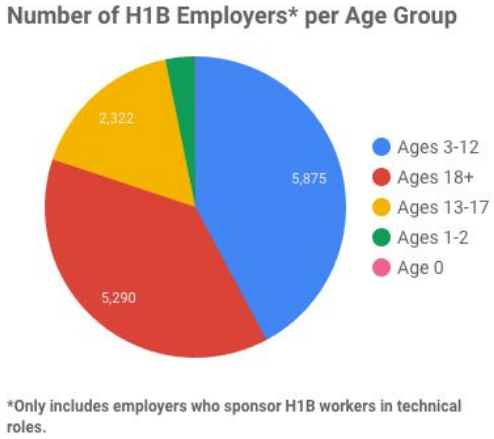
- Assigns a label to the employer based on their age range (0, 1-2, 3-12, 13-17, 18+)

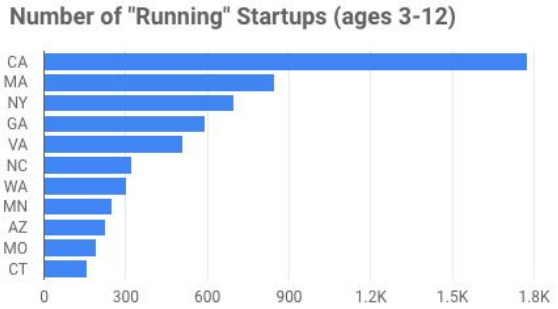
v_Tech_Employer_Age_Label_report:

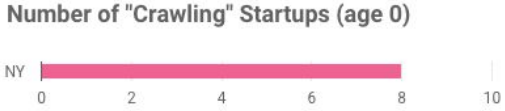
- Groups employers by age label and state combination
- Calculates employer count per group

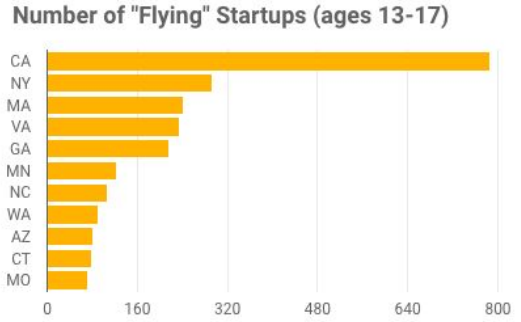
Source File: https://github.com/shirleycohen/h1b_analytics/blob/master/employer_views.sql

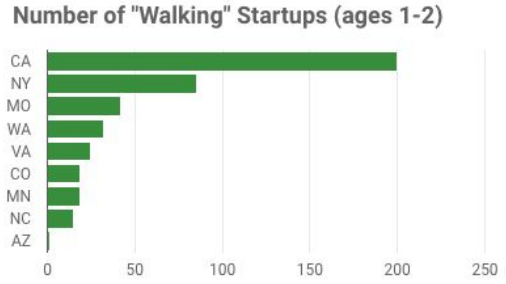
Data Studio Report

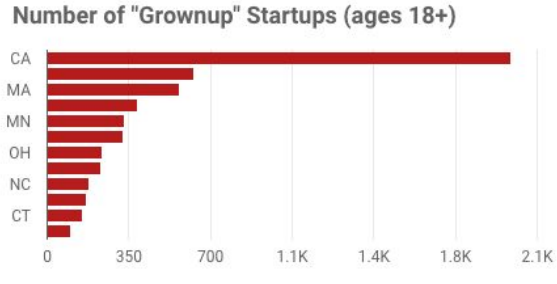












Tips & Tricks

- Always unit test a job on CloudShell before running the same job on Dataflow.
- After each run, review and delete job output logs on CloudShell.
- If writing code locally, delete old code on CloudShell before uploading new code to prevent file renaming.
- If you have a long DoFn, use `print()` to debug DirectRunner job; use `logging.info()` to debug Dataflow job.
- When working with GroupByKey, cast the `UnwindowedValues` object returned to a list in order to iterate through the values.
- When debugging, try to simplify the logic in order to get to the root cause. Error messages can be cryptic and misleading.
- If you've simplified the code and still can't pinpoint the issue, ask for help by providing all the details (including failed experiments) and allow enough time for debugging.

Milestone 8

http://www.cs.utexas.edu/~scohen/milestones/Milestone8.pdf