Class 5 BigQuery

Elements of Databases

Sep 22, 2023

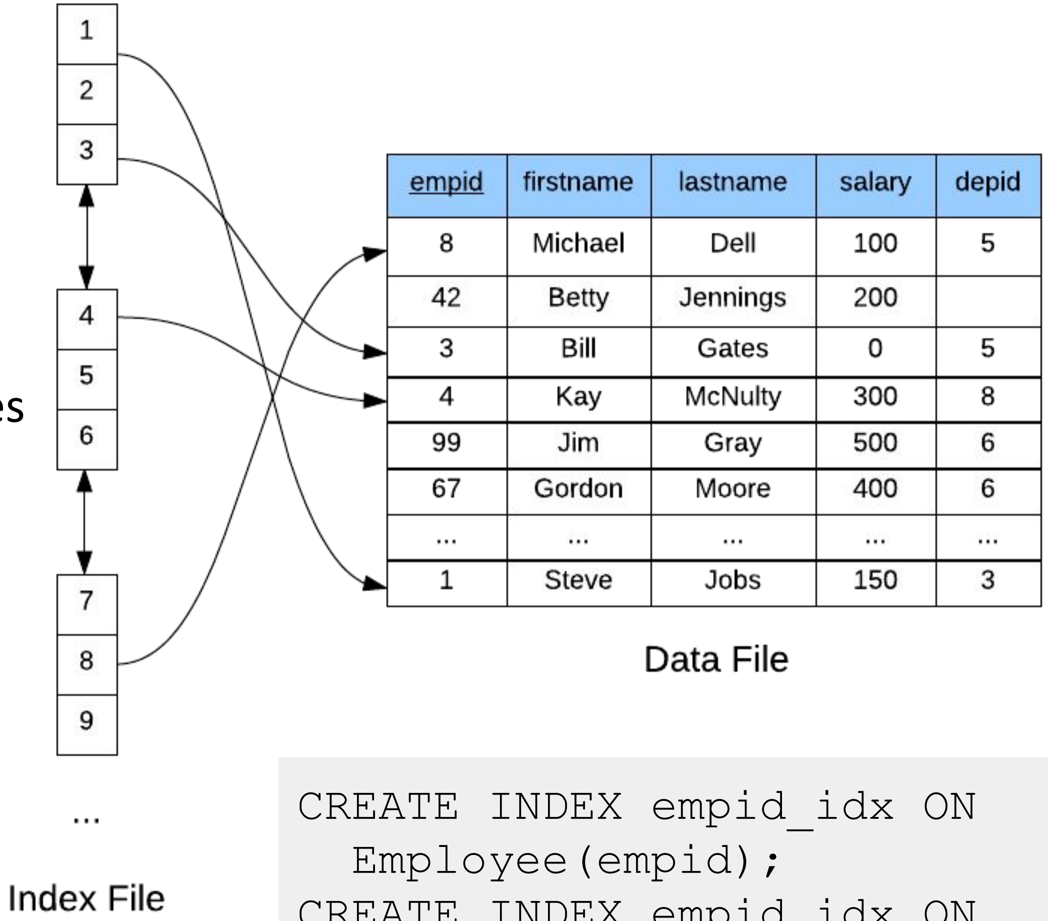
Announcements

Midterm 1:

- Review session: Tues Oct. 3rd at 3pm
- Exam date: Fri Oct 6th at 2pm
- Exam location: WEL 1.316
- Exam will be done on paper. No laptops allowed.
- Closed book and notes. One cheat sheet allowed.

Database Indexes

- **Critical** for many databases
- At least one index per table
- DBA analyzes workload and chooses which indexes to create (no easy answers)
- Creating indexes can be an expensive operation
- They work “behind the scenes”
- Query optimizer decides which indexes to use during execution

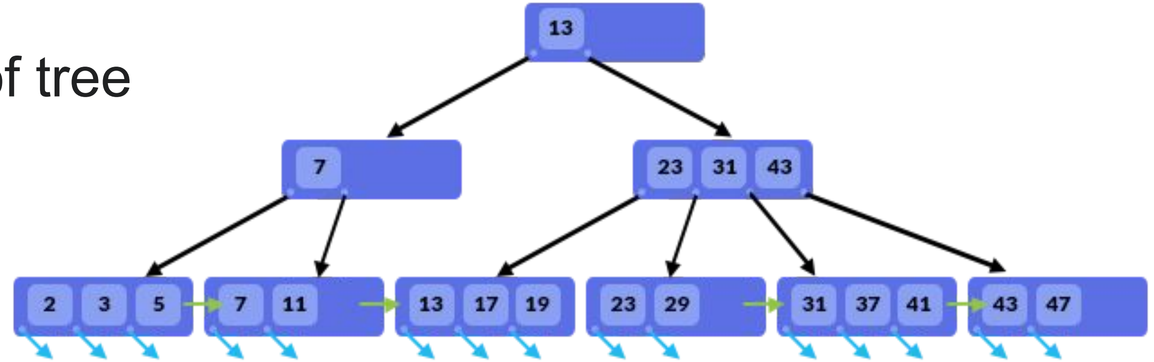


empid	firstname	lastname	salary	depid
8	Michael	Dell	100	5
42	Betty	Jennings	200	
3	Bill	Gates	0	5
4	Kay	McNulty	300	8
99	Jim	Gray	500	6
67	Gordon	Moore	400	6
...	...	...	...	...
1	Steve	Jobs	150	3

```
CREATE INDEX empid_idx ON
  Employee(empid);
CREATE INDEX empid_idx ON
  Employee(empid, salary);
```

B-Trees

- Standard index implementation in most database systems
- Designed to speed up equality and range queries (aka lookups)
- One tree node maps to one disk page (4KB - 8KB)
- A node is packed with index entries (typically 100+)
- Index entry = (key, reference)
- High branching factor = references to child nodes (100+)
- Search speed ≈ height of tree
- Height of tree is O(log *n*) where *n* = index keys



Postgres Code Lab, Part 3

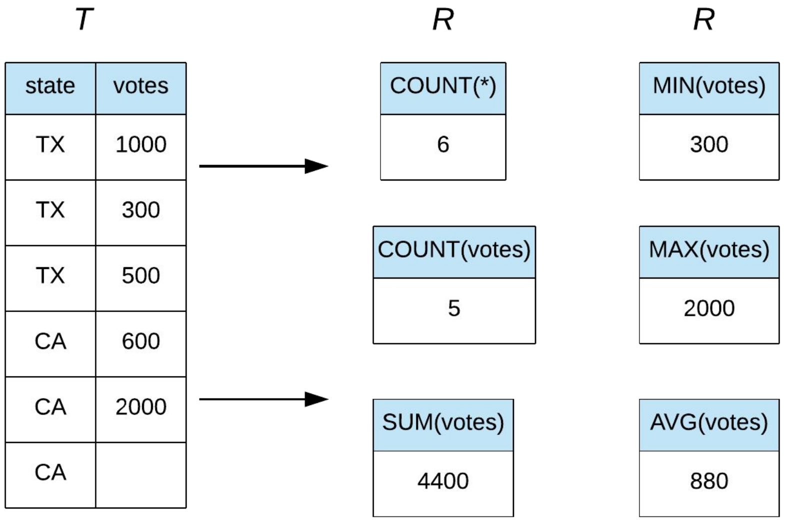
- Clone snippets repo
- Open postgres idx notebook
- Generate an explain plan
- Use \timing
- Create an index to speed up a query

Global Aggregate Queries

```
SELECT <aggregate function>
      [, <aggregate function>]
FROM <single table>
[JOIN <single table>
 ON <join condition>]
[WHERE <boolean condition>]
~~ORDER BY <field(s) to sort on>~~
```

Global Aggregate Queries

```
SELECT <aggregate function>
      [, <aggregate function>]
FROM <single table>
[JOIN <single table>
 ON <join condition>]
[WHERE <boolean condition>]
~~ORDER BY <field(s) to sort on>~~
```



T

state	votes
TX	1000
TX	300
TX	500
CA	600
CA	2000
CA	

R

COUNT(*)
6

COUNT(votes)
5

SUM(votes)
4400

R

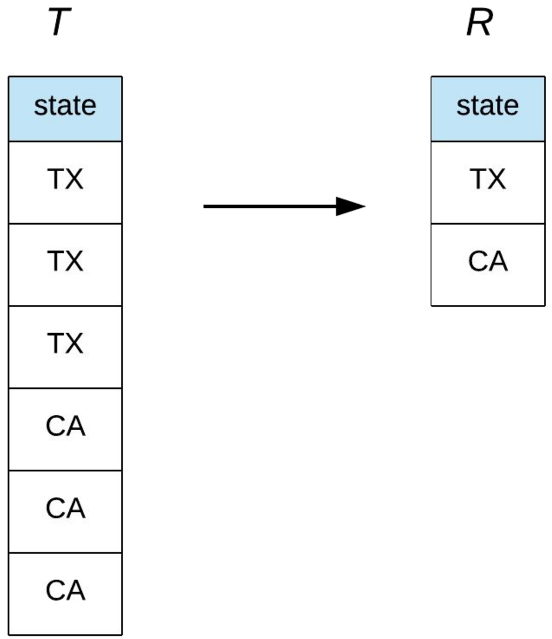
MIN(votes)
300

MAX(votes)
2000

AVG(votes)
880

Group By Queries

```
SELECT **<unaggregated field(s)>**
FROM <single table>
[JOIN <single table>
ON <join condition>]
[WHERE <boolean condition>]
**GROUP BY <unaggregated field(s)>**
```

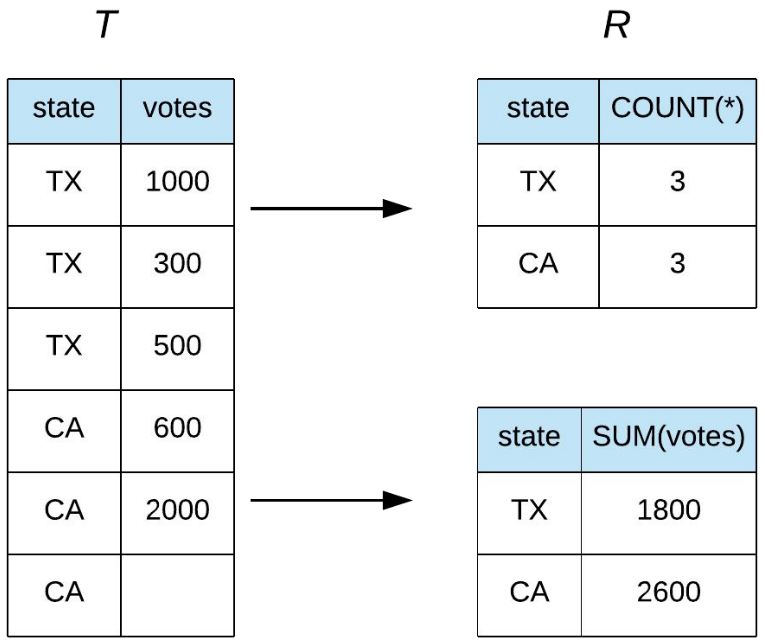


Aggregate Group By Queries

```
SELECT <unaggregated field(s)>,
       <aggregate function(s)>
FROM <single table>
[JOIN <single table>
 ON <join condition>]
[WHERE <boolean condition>]
GROUP BY <unaggregated field(s)>
[HAVING <boolean condition>]
[ORDER BY <field(s) to sort on>]
```

Aggregate Group By Queries

```
SELECT <unaggregated field(s)>,
       <aggregate function(s)>
FROM <single table>
[JOIN <single table>
 ON <join condition>]
[WHERE <boolean condition>]
GROUP BY <unaggregated field(s)>
[HAVING <boolean condition>]
[ORDER BY <field(s) to sort on>]
```



state	votes
TX	1000
TX	300
TX	500
CA	600
CA	2000
CA	

state	COUNT(*)
TX	3
CA	3

state	SUM(votes)
TX	1800
CA	2600

The semantics of `COUNT()`

```
SELECT COUNT(*)
FROM Employee

SELECT COUNT(department)
FROM Employee

SELECT DISTINCT department
FROM Employee

SELECT COUNT(DISTINCT department)
FROM Employee
```

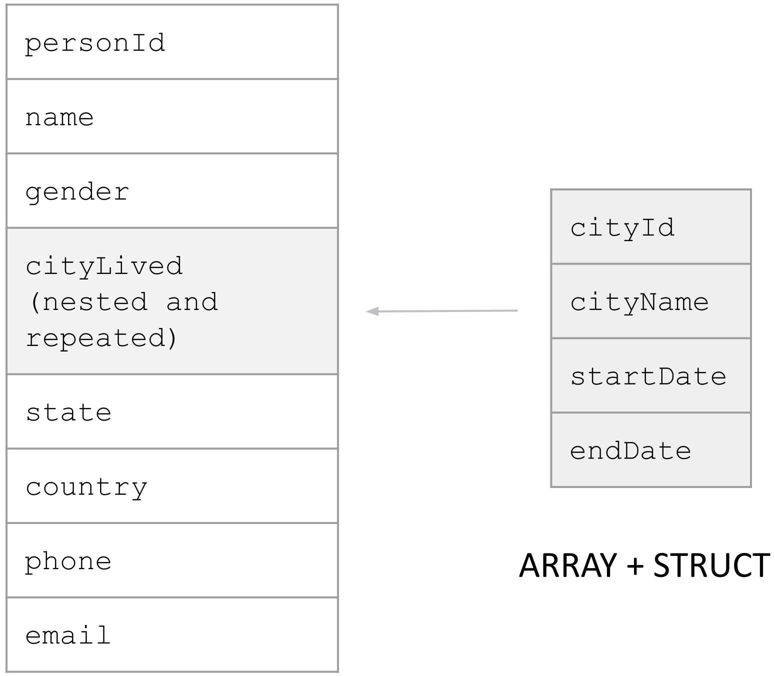
Employee

row	employee	department
1	Sunil	ENG
2	Morgan	ENG
3	Rama	Product
4	Drew	
5	Jeff	Research
6	Danielle	HR
7	Grace	ENG

BigQuery Overview

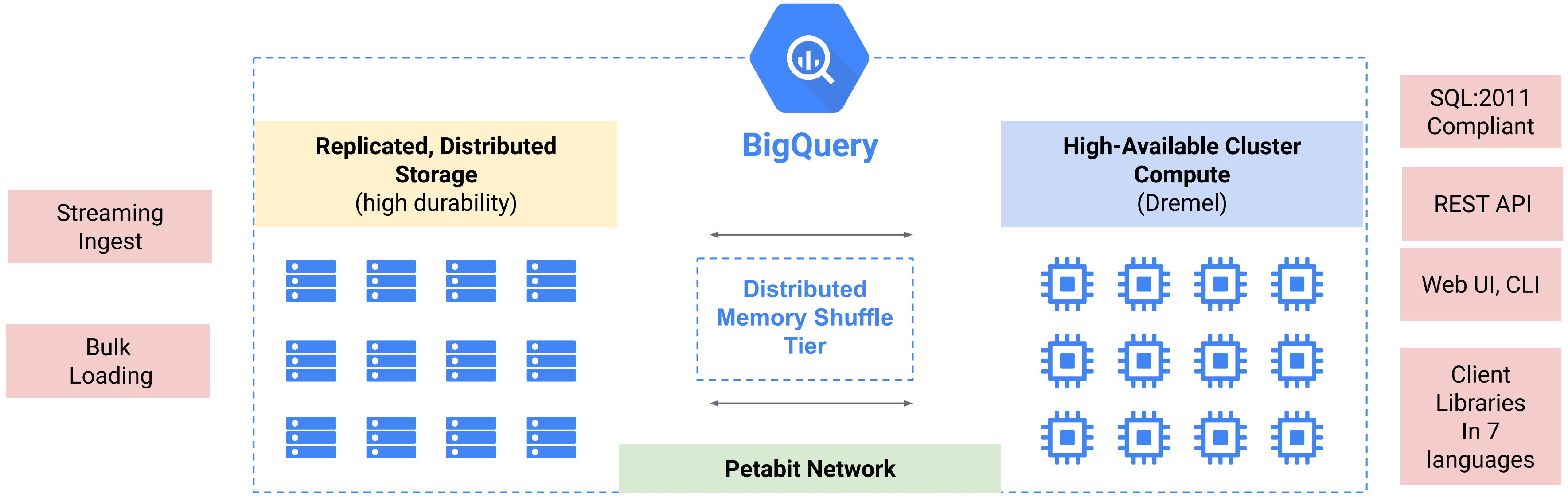
- Data warehouse / analytics database service
- Distributed database system
- Optimized for large data (petabyte-scale)
- Data model: tables with optional nesting
- Query language: standard SQL
- Data Types:
  - Primitive: BOOL, BYTES, FLOAT64, INT64, NUMERIC, STRING
  - Temporal: DATE, DATETIME, TIME, TIMESTAMP
  - Geospatial: GEOGRAPHY
  - Complex: ARRAY, STRUCT
- No provisioning needed, easy to use
- Not an operational database, no referential integrity

Nested Columns



ARRAY + STRUCT type

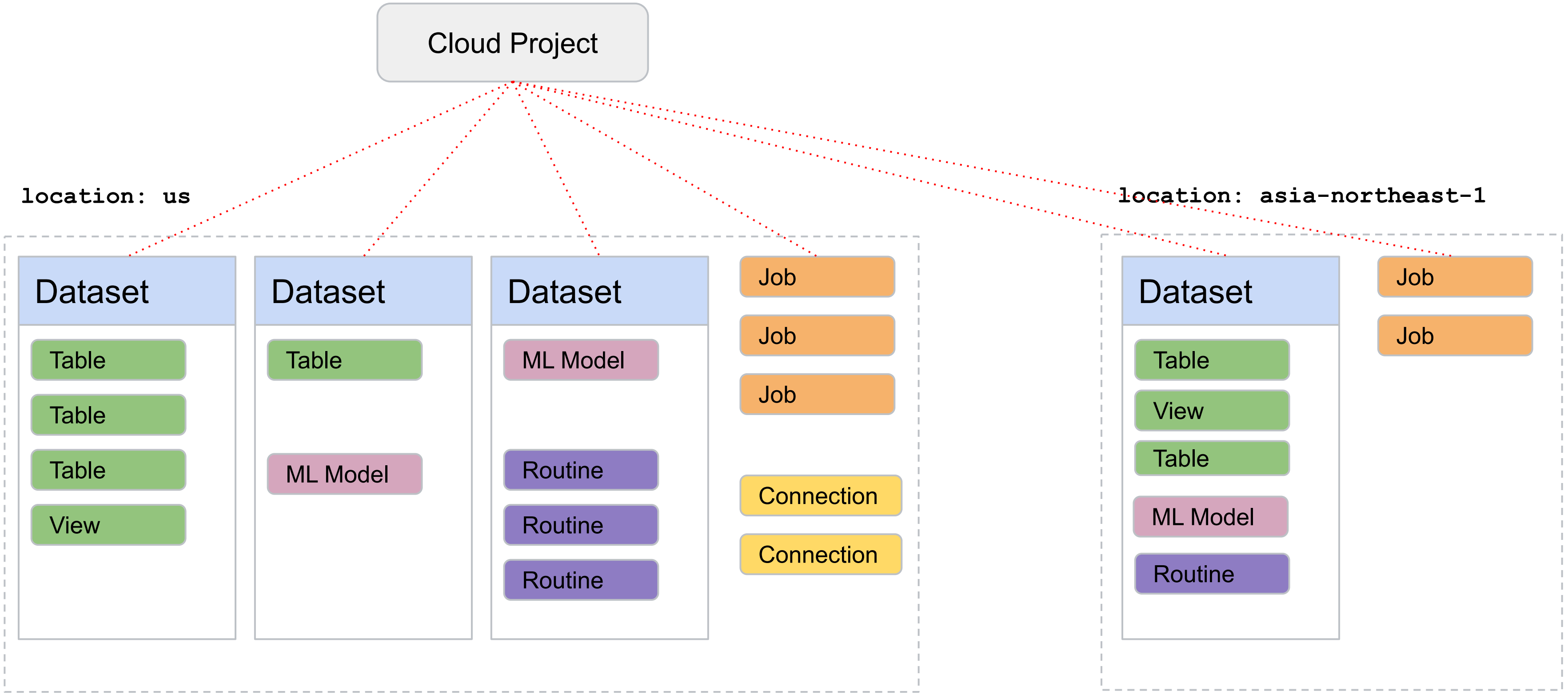
BQ Architecture*



Streaming Ingest
Bulk Loading
Replicated, Distributed Storage
(high durability)
BigQuery
Distributed Memory Shuffle Tier
Petabit Network
High-Available Cluster Compute
(Dremel)
SQL:2011 Compliant
REST API
Web UI, CLI
Client Libraries In 7 languages

* Very approximate

Resource Model



BigQuery Code Lab

- Clone snippets repo
- Open bigquery notebook
- Create college dataset
- Populate college tables
- Explore the data
- Write aggregate queries

Exercise 1: Group-by queries

For each class in the database, obtain the number of students taking the class.

Return the cno for the class along with its enrollment count.

Sort the results by enrollment in descending order.

Database Schema:

Student(sid, fname, lname, dob,
status)

Class(cno, cname, credits)

Instructor(tid, name, dept)

Takes(sid, cno, grade)

Teaches(tid, cno)

Exercise 2: Group-by queries

For each class in the database which has at least two students enrolled, how many students are taking the class?

Return the cno for the class along with its enrollment count.

Sort the results by enrollment in descending order.

Database Schema:

Student(<u>sid</u>, fname, lname, dob, status)

Class(<u>cno</u>, cname, credits)

Instructor(<u>tid</u>, name, dept)

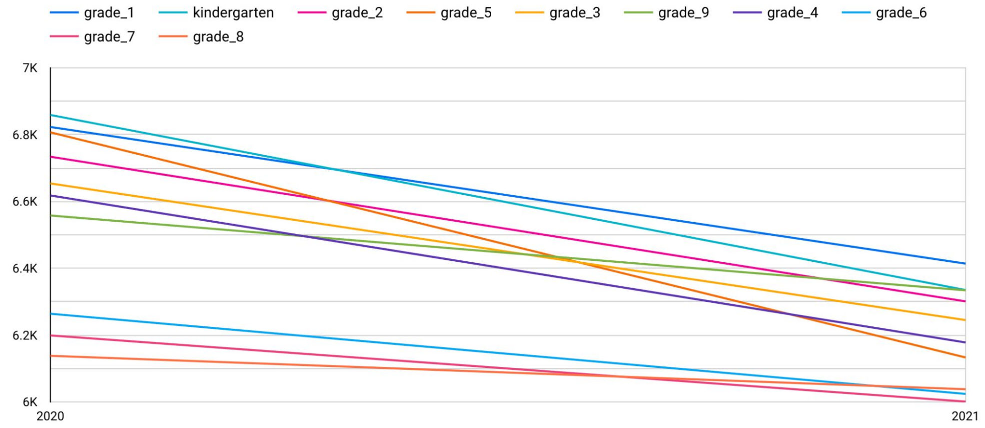
Takes(<u>sid</u>, <u>cno</u>, grade)

Teaches(<u>tid</u>, <u>cno</u>)

Project 4: explore K-12 enrollment data

[Assignment sheet]

- Open [project4 notebook]
- Run starter code:
    - Create and populate tables
    - Explore the data
    - Write aggregate query
    - Create database view
    - Created data visualization


grade_1
kindergarten
grade_2
grade_5
grade_3
grade_9
grade_4
grade_6
grade_7
grade_8
7K
6.8K
6.6K
6.4K
6.2K
6K