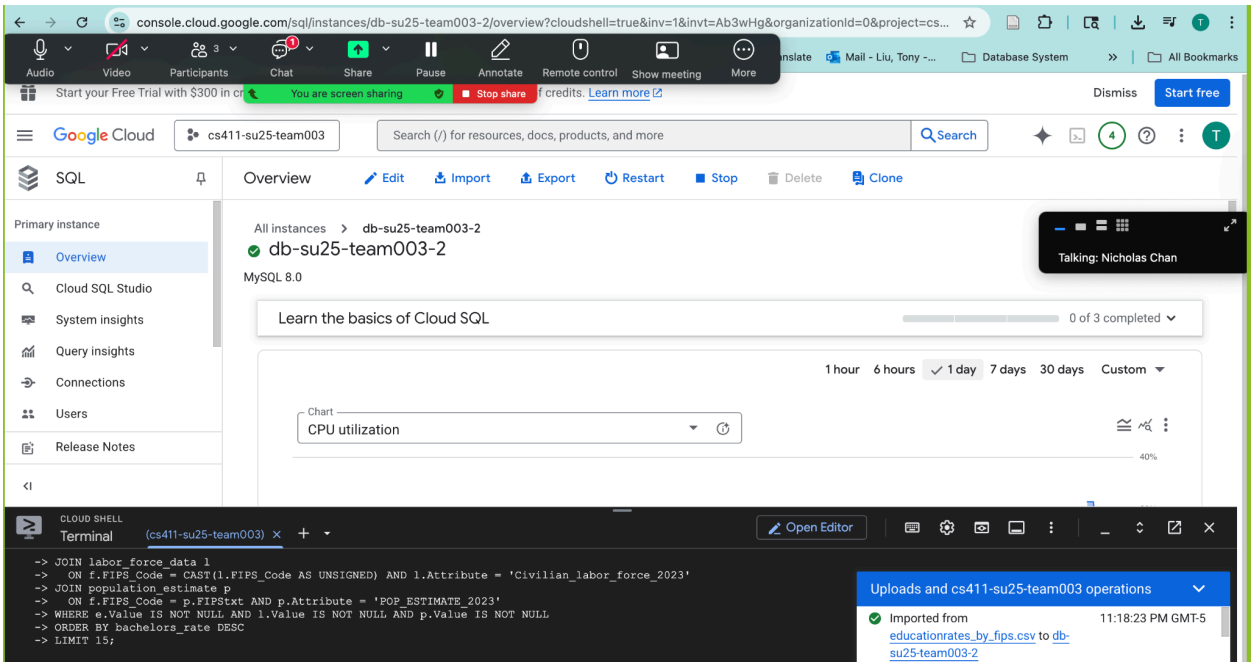
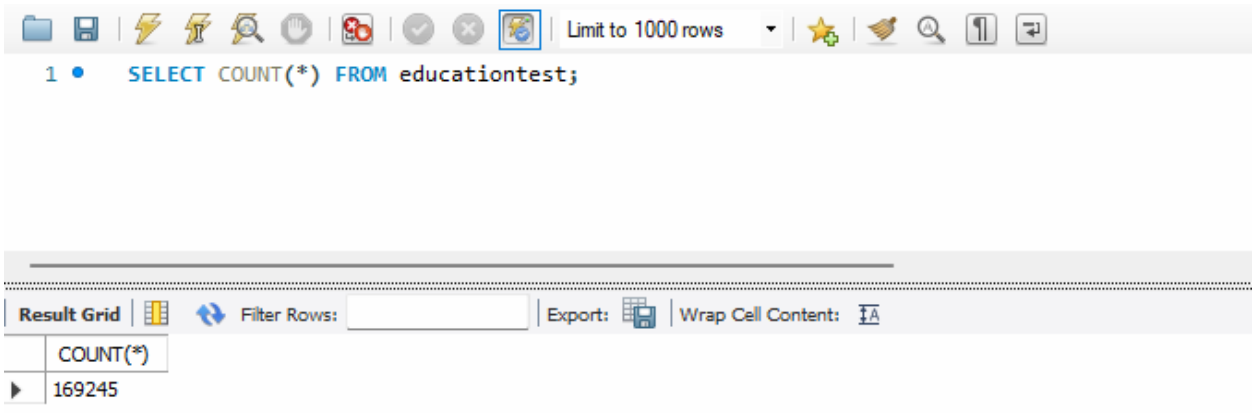
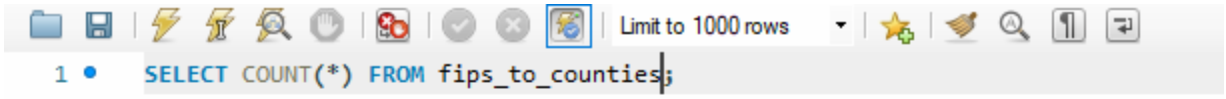
Part 1:

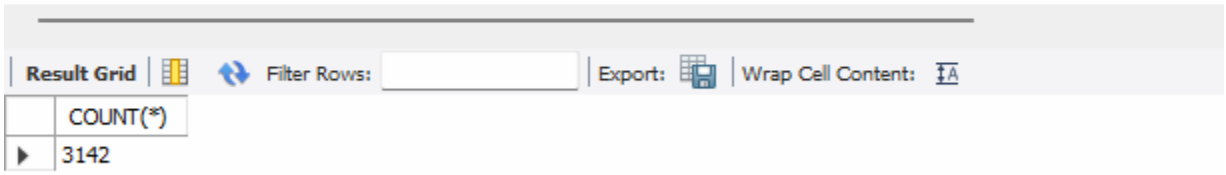
1. Connection on GCP

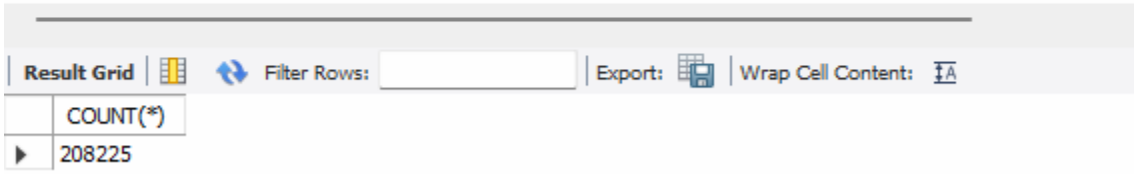


3. Line counts of CSV datasets









4. SQL Advanced Queries

1. ***/Total Labor Force per county:/***

```
SELECT(
  f.County_Name,
  SUM(l.Value) AS total_labor_force))
FROM(
  labor_force_data l)
JOIN(
  fips_to_counties f
    ON l.FIPS_Code = CAST(f.FIPS_Code AS CHAR))
WHERE(
  l.Attribute = 'Civilian_Labor_Force_2023')
```

```
GROUP BY(
  f.County_Name)
ORDER BY
  total_labor_force DESC
LIMIT 15;
```

---

2. **/*CHECK ABOVE NATIONAL AVERAGE BIRTH RATE*/**

```
SELECT DISTINCT
            f.County_Name,
            p.Value AS AboveAvgBirthRate
FROM
           population_estimate p
JOIN
           fips_to_counties f ON p.FIPStxt = f.FIPS_Code
WHERE
  p.Attribute LIKE '%BIRTHS_2023%'
  AND p.Value > (
    SELECT AVG(Value)
    FROM population_estimate
    WHERE Attribute LIKE '%BIRTHS_2023%'
  )
ORDER BY
  p.Value DESC
LIMIT 15;
```

---

3. **/* Socioeconomic Data per State */**

```
SELECT
  f.County_Name,
  f.State,
  p.Value AS population_2023,
  l.Value AS labor_force_2023,
  g.percentChange2023 AS gdp_change_2023,
  e.Value AS education_metric
FROM fips_to_counties f
LEFT JOIN population_estimate p
  ON f.FIPS_Code = p.FIPStxt AND p.Attribute = 'POP_ESTIMATE_2023'
LEFT JOIN labor_force_data l
```

```
  ON f.FIPS_Code = CAST(l.FIPS_Code AS UNSIGNED) AND l.Attribute =
'Civilian_labor_force_2023'
LEFT JOIN gdp_by_county g
  ON f.County_Name = g.countyName
LEFT JOIN educationtest e
  ON f.FIPS_Code = e.FIPSCode AND e.Attribute = 'Bachelor\'s degree or higher; 2008-12'
WHERE f.State = 'Alabama'
LIMIT 15;
```

---

4. **/*Calculating bachelors_rate: abundant employment opportunity*/**

```
SELECT
  f.County_Name,
  f.State,
  e.Value AS bachelors_rate,
  l.Value AS labor_force,
  ROUND(l.Value / p.Value * 100, 2) AS labor_force_participation_pct
FROM fips_to_counties f
JOIN educationtest e
  ON f.FIPS_Code = e.FIPSCode AND e.Attribute = "Bachelor's degree or higher; 2008-12"
JOIN labor_force_data l
  ON f.FIPS_Code = CAST(l.FIPS_Code AS UNSIGNED) AND l.Attribute =
'Civilian_labor_force_2023'
JOIN population_estimate p
  ON f.FIPS_Code = p.FIPStxt AND p.Attribute = 'POP_ESTIMATE_2023'
WHERE e.Value IS NOT NULL AND l.Value IS NOT NULL AND p.Value IS NOT NULL
ORDER BY bachelors_rate DESC
LIMIT 15;
```

5. SQL PROOF

CHECK States ABOVE NATIONAL AVERAGE BIRTH RATE

County_Name	AboveAvgBirthRate
Los Angeles	95354
Harris	65450
Cook	53124
Maricopa	50795
San Diego	37145
Dallas	36886
Kings	31066
Orange	30677
Miami-Dade	29339
Tarrant	28006
Riverside	27497
Bexar	26765
San Bernardino	26188
Clark	24876
Queens	23304

Socioeconomic Data per State

County_Name	State	population_2023	labor_force_2023	gdp_change_2023	education_metric
Autauga	Alabama	60342	27000	2.6	7629
Baldwin	Alabama	253507	104409	3.3	35307
Baldwin	Alabama	253507	104409	3.2	35307
Barbour	Alabama	24585	7944	0	2757
Barbour	Alabama	24585	7944	-3.9	2757
Bibb	Alabama	21868	8772	-0.6	1396
Bibb	Alabama	21868	8772	1.9	1396
Blount	Alabama	59816	25836	4	4792
Blount	Alabama	59816	25836	-4.3	4792
Bullock	Alabama	9897	4500	6.1	900
Butler	Alabama	18382	9009	5.6	1812
Butler	Alabama	18382	9009	2.1	1812
Butler	Alabama	18382	9009	10.1	1812
Butler	Alabama	18382	9009	2.2	1812
Butler	Alabama	18382	9009	4.5	1812

Total Labor Force per county

County_Name	total_labor_force
Los Angeles	5015629
Orange	2763772
Cook	2663106
Maricopa	2441028
Harris	2431621
Montgomery	1936650
Jefferson	1828596
Clark	1618537
San Diego	1596432
Dallas	1539843
Middlesex	1483888
Washington	1397063
Miami-Dade	1377572
King	1349289
Kings	1283772

Bachelors Rate

County_Name	State	bachelors_rate	labor_force	labor_force_participation_pct
Los Angeles	California	1879673	5015629	51.9
Cook	Illinois	1184199	2651912	52.13
Orange	California	723978	1588873	50.67
Maricopa	Arizona	722479	2441028	53.23
Harris	Texas	713494	2414902	49.94
San Diego	California	691991	1596432	48.82
New York	New York	687519	935314	58.55
King	Washington	619908	1348792	59.38
Santa Clara	California	552652	1038951	55.33
Middlesex	Massachusetts	523916	915035	56.35
Kings	New York	492484	1225540	47.85
Queens	New York	466251	1153534	51.22
Miami-Dade	Florida	452145	1377572	51.27
Fairfax	Virginia	429065	656494	57.49
Alameda	California	421610	826102	50.93

Part 2:

1. Index Analysis on The First Query:
    a. Table overview:

```
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| FIPS_Code | int          | NO   | PRI | NULL    |       |
| State     | varchar(5)   | YES  |     | NULL    |       |
| Area_Name | varchar(255) | YES  |     | NULL    |       |
| Attribute | varchar(100) | NO   | PRI | NULL    |       |
| Value     | double       | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

    b. Performance Analysis on Indexing Attributes

Index Added	Query Cost (Estimated)	Access Type	Notes
None (baseline)	N/A (not shown here)	ALL, Nested Loop Join	Full table scans with unindexed filter
idx_labor_force_attribute	cost=3557	Single-row index lookup on l using PRIMARY	Reduced filtering cost using Attribute index
idx_labor_force_fips	cost=3566	Single-row index lookup	JOIN speed improved, but cost slightly higher
idx_fips_to_county	cost=3170	Covering index scan on f	Significant improvement in JOIN using county name

2. Index Analysis On The Second Query:
    a. Table Overview:

```
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| FIPStxt   | int          | NO   | PRI | NULL    |       |
| State     | varchar(2)   | YES  |     | NULL    |       |
| Area_Name | varchar(100) | YES  |     | NULL    |       |
| Attribute | varchar(100) | NO   | PRI | NULL    |       |
```

```
| Value     | double       | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

b. Performance Analysis on Indexing Attributes

Index Added	Query Cost (estimated)	Observed Behavior	Notes
None (baseline)	*(approx.) cost ≈ 23582*	Full scan on population_estimate, subquery unoptimized	Baseline omitted but inferred from first EXPLAIN with idx_population_attr
idx_population_attr	cost=23582	Minor improvement; filter on Attribute accelerated	Helped outer and subquery filtering, but JOIN still not optimized
idx_population_fipstxt	cost=18790	JOIN optimized; faster FIPStxt → FIPS_Code mapping	Helped JOIN but subquery still full scan on Attribute
idx_population_combo	cost=22874	Mixed improvement; JOIN and outer filter faster, subquery not benefited	Could not accelerate subquery; LIKE on Attribute hard to optimize

3. Index Analysis On The Third Query:
   a. Table Overview

```
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| State       | varchar(50) | YES  |     | NULL    |       |
| County_Name | varchar(50) | YES  | MUL | NULL    |       |
| FIPS_Code   | int         | NO   | PRI | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

   b. Performance Analysis on Indexing Attributes

Step	Action	Index	Cost (from EXPLAIN)	Actual Time	Notes
1	Baseline (no index)	—	**201779**	~0.55 sec	High cost; full scans on filter columns

2	Create index on population_estimate(Attribute)	idx_pop_attribute	201779 (same)	Slightly better ~0.36 sec	Marginal improvement; joins still slow
3	Create composite index on labor_force_data(FIPS_Code, Attribute)	idx_labor_force_attribute	↓ to **198289**	~0.04 sec	**Significant drop in cost and runtime**
4	Create composite index on educationtest(FIPSCode, Attribute)	idx_edu_composite	198289 (no change from Step 3)	~0.04 sec	Retains performance gain; joins on educationtest now efficient

4. Index Analysis On The Fourth Query:

    a. Performance Analysis on Indexing Attributes

Index Added	Query Cost (Estimated)	Access Type	Notes
None (baseline)	N/A (not shown here)	ALL, Nested Loop Join	Full table scans with unindexed filter
idx_labor_force_attribute	cost=3557	Single-row index lookup on l using PRIMARY	Reduced filtering cost using Attribute index
idx_labor_force_fips	cost=3566	Single-row index lookup	JOIN speed improved, but cost slightly higher
idx_fips_to_county	cost=3170	Covering index scan on f	Significant improvement in JOIN using county name

The final indexing design we chose was covering index scan on f because it in many situations significantly reduced the query cost due to the construction of our database. Our database is made up of many different tables due to our need for conversion tables to change from one variable type to another. This constant need to reference other tables with join statements significantly increases our query cost and further slows down our processes due to the large amounts of data. Choosing an access type that significantly improves the join time helps our program function at a much better rate which is essential when dealing with such large data sets. We did see a difference in our results due to the overall size of the data so there was a clear best option.

Bachelor's Rate

Before INDEXING

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=350..350 rows=15 loops=1)
  -> Sort: e.`Value` DESC, limit input to 15 row(s) per chunk  (actual time=350..350 rows=15 loops=1)
    -> Stream results  (cost=16.9e+6 rows=14.4e+6) (actual time=182..350 rows=3124 loops=1)
      -> Inner hash join (p.FIPStxt = cast(l.FIPS_Code as unsigned))  (cost=16.9e+6 rows=14.4e+6) (actual time=182..349
```

CREATE INDEX idx_education_attribute ON educationtest (Attribute);
CREATE INDEX idx_labor_attribute ON labor_force_data (Attribute);
CREATE INDEX idx_population_attribute ON population_estimate (Attribute);

AFTER INDEXING

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=10895..10895 rows=15 loops=1)
  -> Sort: e.`Value` DESC, limit input to 15 row(s) per chunk  (actual time=10895..10895 rows=15 loops=1)
    -> Stream results  (cost=6.14e+6 rows=1.6e+6) (actual time=0.147..10893 rows=3124 loops=1)
      -> Nested loop inner join  (cost=6.14e+6 rows=1.6e+6) (actual time=0.143..10887 rows=3124 loops=1)
```

These queries use conditions such as e.attribute, l.attribute, and p.attribute in where/join clauses. Creating indexes allowed the database to quickly find the rows without scanning the entire table. This is why the cost got lower.