CS 487/587 Database Implementation
Winter 2021
Database Benchmarking Project
Part I - Data Generation & System Selection

Likhitha Vanga
Jaya Bhargavi Vengala

We chose option-1 for the project.

Option 1: System Comparison: Compare two systems (on the same hardware). Compare the performance of two different 'relational' systems by running queries on both systems and comparing the performance

System-1: PostgreSQL

We used Postgresql**.**We chose this because postgresql is an object relational database and can handle features like table inheritance and function overloading, which can be important to certain applications. Postgres handles concurrency very well. Postgres is known for protecting data integrity at the transaction level and this makes it less vulnerable to data corruption.
Also, we wanted this as an opportunity to learn more about postgresql and gain expertise in this database.

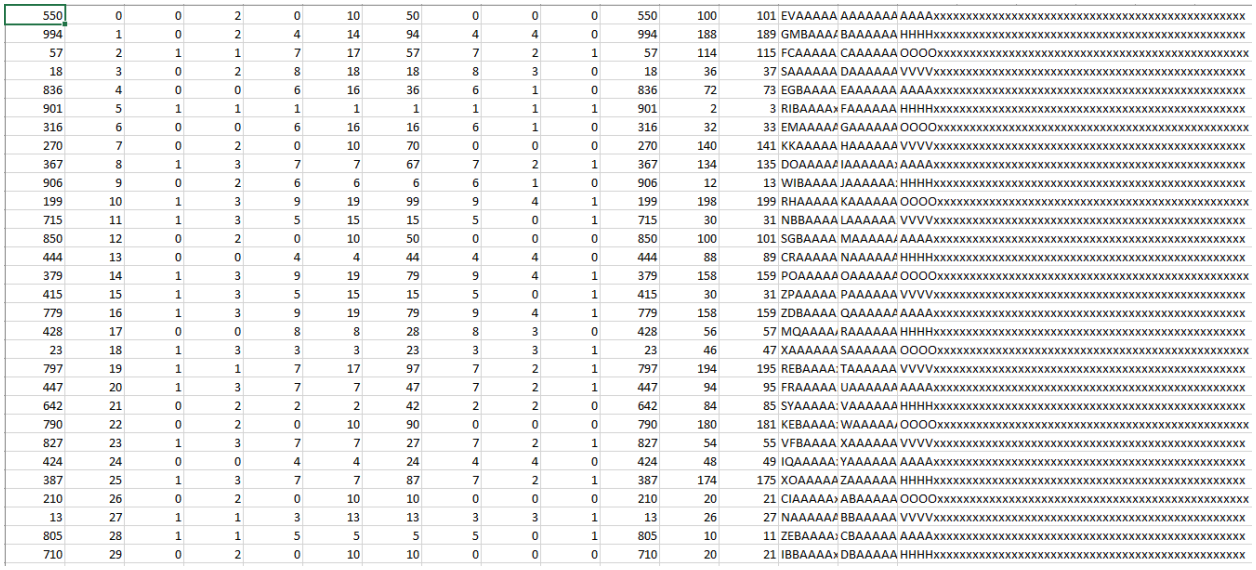
System-2: Big Query

BigQuery is Google's serverless data warehouse. BigQuery's "serverless" build, a fully on-cloud design that prioritizes scalability and quickness in queries, means that you can easily scale and perform ad hoc analyses much faster than you would on cloud-based server structures. Even better, its decentralized design means it can perform these queries and derive insights from datasets that stretch into petabyte scale. It solves this problem by enabling super-fast SQL queries using the processing power of Google's infrastructure.

We want to investigate these two systems as one is cloud based. There are a lot of similarities between these 2 systems yet make them different based on their usage and difference in core concepts.

Brief description of your data generation process including what program you used to generate the data.

We have used a program of our own to generate onektup and 2 tenktup csv files.



550	0	0	2	0	10	50	0	0	0	550	100	101	EVAAAAA	AAAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
994	1	0	2	4	14	94	4	4	0	994	188	189	GMBAAAA	BAAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
57	2	1	1	7	17	57	7	2	1	57	114	115	FCAAAAA	CAAAAAA	OOOOxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
18	3	0	2	8	18	18	8	3	0	18	36	37	SAAAAAA	DAAAAAA	VVVVxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
836	4	0	0	6	16	36	6	1	0	836	72	73	EGBAAAA	EAAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
901	5	1	1	1	1	1	1	1	1	901	2	3	RIBAAAA	FAAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
316	6	0	0	6	16	16	6	1	0	316	32	33	EMAAAAA	GAAAAAA	OOOOxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
270	7	0	2	0	10	70	0	0	0	270	140	141	KKAAAAA	HAAAAAA	VVVVxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
367	8	1	3	7	7	67	7	2	1	367	134	135	DOAAAAA	IAAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
906	9	0	2	6	6	6	6	1	0	906	12	13	WIBAAAA	JAAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
199	10	1	3	9	19	99	9	4	1	199	198	199	RHAAAAA	KAAAAAA	OOOOxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
715	11	1	3	5	15	15	5	0	1	715	30	31	NBBAAAA	LAAAAAA	VVVVxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
850	12	0	2	0	10	50	0	0	0	850	100	101	SGBAAAA	MAAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
444	13	0	0	4	4	44	4	4	0	444	88	89	CRAAAAA	NAAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
379	14	1	3	9	19	79	9	4	1	379	158	159	POAAAAA	OAAAAAA	OOOOxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
415	15	1	3	5	15	15	5	0	1	415	30	31	ZPAAAAA	PAAAAAA	VVVVxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
779	16	1	3	9	19	79	9	4	1	779	158	159	ZDBAAAA	QAAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
428	17	0	0	8	8	28	8	3	0	428	56	57	MQAAAAA	RAAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
23	18	1	3	3	3	23	3	3	1	23	46	47	XAAAAAA	SAAAAAA	OOOOxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
797	19	1	1	7	17	97	7	2	1	797	194	195	REBAAAA	TAAAAAA	VVVVxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
447	20	1	3	7	7	47	7	2	1	447	94	95	FRAAAAA	UAAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
642	21	0	2	2	2	42	2	2	0	642	84	85	SYAAAAA	VAAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
790	22	0	2	0	10	90	0	0	0	790	180	181	KEBAAAA	WAAAAAA	OOOOxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
827	23	1	3	7	7	27	7	2	1	827	54	55	VFBAAAA	XAAAAAA	VVVVxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
424	24	0	0	4	4	24	4	4	0	424	48	49	IQAAAAA	YAAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
387	25	1	3	7	7	87	7	2	1	387	174	175	XOAAAAA	ZAAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
210	26	0	2	0	10	10	0	0	0	210	20	21	CIAAAAA	ABAAAAA	OOOOxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
13	27	1	1	3	13	13	3	3	1	13	26	27	NAAAAAA	BBAAAAA	VVVVxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
805	28	1	1	5	5	5	5	0	1	805	10	11	ZEBAAAA	CBAAAAA	AAAAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
710	29	0	2	0	10	10	0	0	0	710	20	21	IBBAAAA	DBAAAAA	HHHHxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Logic used to generate data from wisconsin paper:

As stated in the Wisconsin Benchmark paper, the original benchmark has tree relations:

onkup, tenkup1, tenkup2. There differences are:

- Onekup means there are 1000 tuples on the table.
- Tenkup1 and tenkup2 have 10000 tuples respectively.

Where their schema in the table are as follows:

- 13 integer attributes.
- 3 52-byte string attribute

Onekup.csv, tenkup1.csv and tenkup2.csv are in the data folder and wb.py is the script to generate data

```python
# convert to candidate key according to wisconsin benchmark paper
def convert(uni):
    result = list('A'*7)
    i = 6
    while uni > 0:
        rem = uni % 26
        result[i] = chr(ord('A')+rem)
        uni = uni // 26
        i -= 1
    result.reverse()
    return "".join(result) + "x" * 45

# generate string4
def stringFour(tupCount):
    i = tupCount
    if i % 4 == 0:
        return "A" * 4 + "x" * 48
    elif i % 4 == 1:
        return "H" * 4 + "x" * 48
    elif i % 4 == 2:
        return "O" * 4 + "x" * 48
    else:
        return "V" * 4 + "x" * 48
```

Demonstrate you have loaded data into that system

Schema:

wisconson/postgres@PostgreSQL 11

Query Editor | Query History

```
9  DROP TABLE if exists tenktup2;
10
11 CREATE TABLE tenktup2 (
12     unique1 int NOT NULL,
13     unique2 int PRIMARY KEY NOT NULL,
14     two int NOT NULL,
15     four int NOT NULL,
16     ten int NOT NULL,
17     twenty int NOT NULL,
18     onePercent int NOT NULL,
19     tenPercent int NOT NULL,
20     twentyPercent int NOT NULL,
21     fiftyPercent int NOT NULL,
22     unique3 int NOT NULL,
23     evenOnePercent int NOT NULL,
24     oddOnePercent int NOT NULL,
25     stringu1 varchar(52) NOT NULL,
26     stringu2 varchar(52) NOT NULL,
27     string4 varchar(52) NOT NULL
28   );
29
30 -- ALTER TABLE onektup OWNER to postgres;
31
32 -- COPY onektup(unique1, unique2, two, four, ten, twenty, onePercent, tenPercent, twentyPercent, fiftyPercent, unique3, evenOnePercent, oddOnePercent, stringu1, st
33 -- FROM 'C:\Users\prane\Desktop\Likhitha\Winter-2021\DB Imp\Project\CS-587-DBImplem\data\onektup.csv' DELIMITER ',' HEADER CSV;
34
35 COPY tenktup2(unique1, unique2, two, four, ten, twenty, onePercent, tenPercent, twentyPercent, fiftyPercent, unique3, evenOnePercent, oddOnePercent, stringu1, str
36 FROM 'C:\Users\prane\Desktop\Likhitha\Winter-2021\DB Imp\Project\CS-587-DBImplem\data\tenktup2.csv' DELIMITER ',' HEADER CSV;
```

Onektup table:

	unique1 integer	unique2 [PK] integer	two integer	four integer	ten integer	twenty integer	onepercent integer	tenpercent integer	twentypercent integer	fiftypercent integer	unique3 integer	evenonepercent integer	oddonepercent integer	stringu1 character var
1	994	1	0	2	4	14	94	4	4	0	994	188	189	GMBAAAAxx
2	57	2	1	1	7	17	57	7	2	1	57	114	115	FCAAAAAxx
3	18	3	0	2	8	18	18	8	3	0	18	36	37	SAAAAAAxx
4	836	4	0	0	6	16	36	6	1	0	836	72	73	EGBAAAAxx
5	901	5	1	1	1	1	1	1	1	1	901	2	3	RIBAAAAxx
6	316	6	0	0	6	16	16	6	1	0	316	32	33	EMAAAAAx
7	270	7	0	2	0	10	70	0	0	0	270	140	141	KKAAAAAx
8	367	8	1	3	7	7	67	7	2	1	367	134	135	DOAAAAAx
9	906	9	0	2	6	6	6	6	1	0	906	12	13	WIBAAAAx
10	199	10	1	3	9	19	99	9	4	1	199	198	199	RHAAAAAx
11	715	11	1	3	5	15	15	5	0	1	715	30	31	NBBAAAAx
12	850	12	0	2	0	10	50	0	0	0	850	100	101	SGBAAAAx
13	444	13	0	0	4	4	44	4	4	0	444	88	89	CRAAAAAx
14	379	14	1	3	9	19	79	9	4	1	379	158	159	POAAAAAx
15	415	15	1	3	5	15	15	5	0	1	415	30	31	ZPAAAAAx
16	779	16	1	3	9	19	79	9	4	1	779	158	159	ZDBAAAAx
17	428	17	0	0	8	8	28	8	3	0	428	56	57	MQAAAAAx
18	23	18	1	3	3	3	23	3	3	1	23	46	47	XAAAAAAx
19	797	19	1	1	7	17	97	7	2	1	797	194	195	REBAAAAx
20	447	20	1	3	7	7	47	7	2	1	447	94	95	FRAAAAAx
21	642	21	0	2	2	2	42	2	2	0	642	84	85	SYAAAAAx
22	790	22	0	2	0	10	90	0	0	0	790	180	181	KFBAAAAx

Tenktup table:

Query Editor Query History

```
1  SELECT * FROM public.tenktup1
2  ORDER BY unique2 ASC
```

Data Output Explain Messages Notifications

	unique1 integer	unique2 [PK] integer	two integer	four integer	ten integer	twenty integer	onepercent integer	tenpercent integer	twentypercent integer	fiftypercent integer	unique3 integer	evenonepercent integer	oddonepercent integer	stringu1 character vary
1	4946	1	0	2	6	6	46	6	1	0	4946	92	93	GIHAAAAx
2	6824	2	0	0	4	4	24	4	4	0	6824	48	49	MCKAAAAx
3	320	3	0	0	0	0	20	0	0	0	320	40	41	IMAAAAAx
4	7860	4	0	0	0	0	60	0	0	0	7860	120	121	IQLAAAAx
5	1406	5	0	2	6	6	6	6	1	0	1406	12	13	CCCAAAAx
6	77	6	1	1	7	17	77	7	2	1	77	154	155	ZCAAAAAx
7	7637	7	1	1	7	17	37	7	2	1	7637	74	75	THLAAAAx
8	6809	8	1	1	9	9	9	9	4	1	6809	18	19	XBKAAAAx
9	1842	9	0	2	2	2	42	2	2	0	1842	84	85	WSCAAAAx
10	9949	10	1	1	9	9	49	9	4	1	9949	98	99	RSOAAAAx
11	5503	11	1	3	3	3	3	3	3	1	5503	6	7	RDIAAAAx
12	1565	12	1	1	5	5	65	5	0	1	1565	130	131	FICAAAAx
13	308	13	0	0	8	8	8	8	3	0	308	16	17	WLAAAAAx
14	8306	14	0	2	6	6	6	6	1	0	8306	12	13	MHMAAAAx
15	454	15	0	2	4	14	54	4	4	0	454	108	109	MRAAAAAx
16	3645	16	1	1	5	5	45	5	0	1	3645	90	91	FKFAAAAx
17	4604	17	0	0	4	4	4	4	4	0	4604	8	9	CVGAAAAx
18	8562	18	0	2	2	2	62	2	2	0	8562	124	125	IRMAAAAx
19	2020	19	0	0	0	0	20	0	0	0	2020	40	41	SZCAAAAx
20	4625	20	1	1	5	5	25	5	0	1	4625	50	51	XVGAAAAx
21	3624	21	0	0	4	4	24	4	4	0	3624	48	49	KJFAAAAx
22	9859	22	1	3	9	19	59	9	4	1	9859	118	119	FBQAAAAx

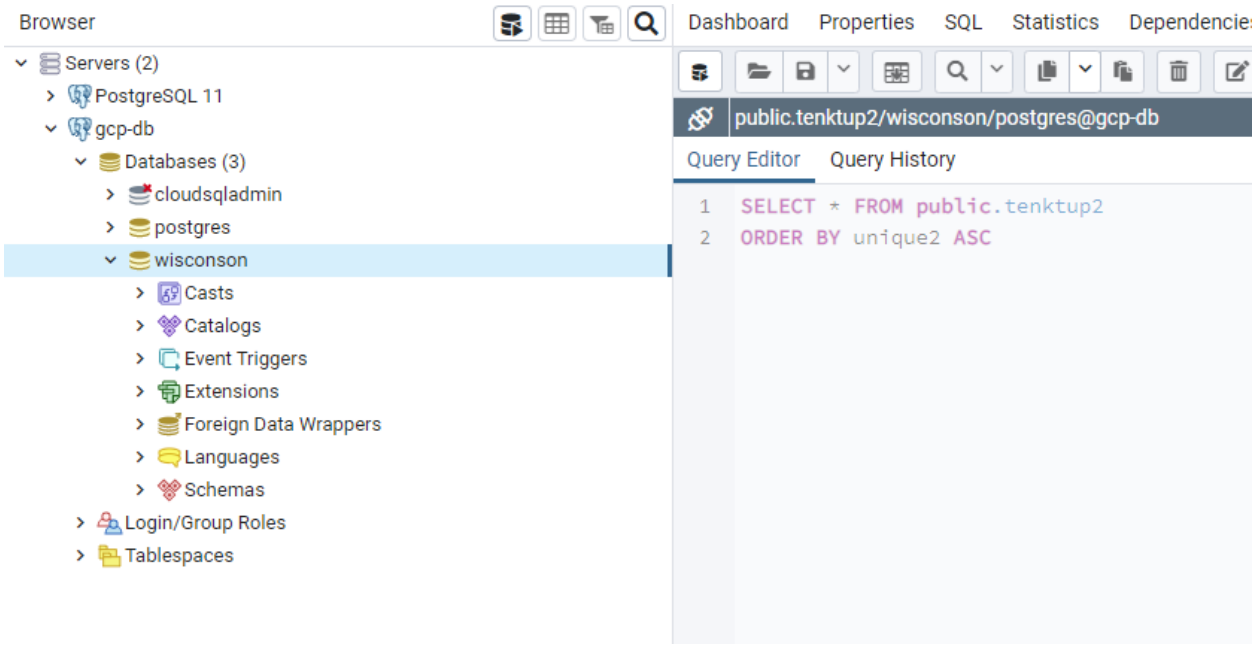
Include lessons learned or issues encountered:

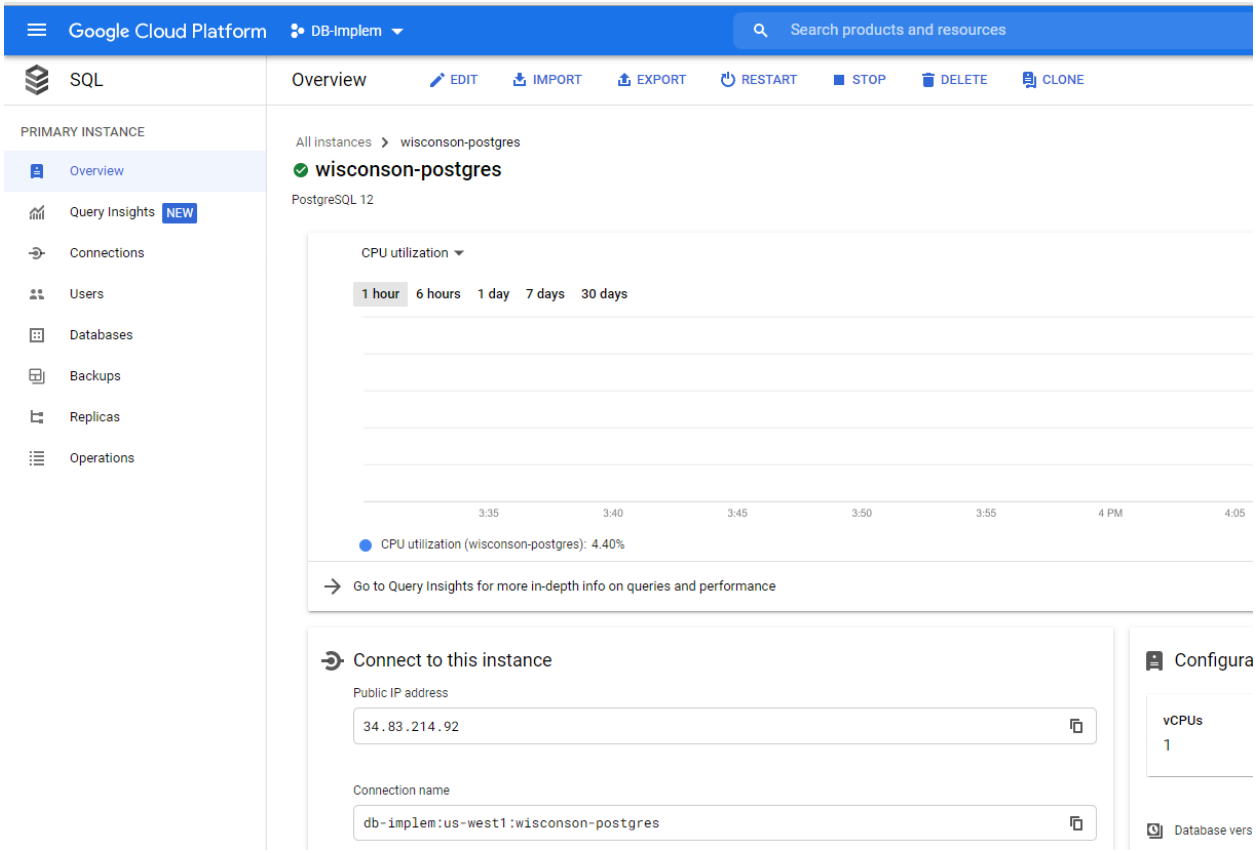
- We found that python has libraries for postgres integration.
- Understood the creation of postgres database on GCP.

- Understood the server connection from local to GCP.
- To load a CSV file to postgres, we need permission from the file access.
- Loading a csv file from local to big query.
- Understood Wisconsin benchmark design.
- Resolved an issue regarding connection failure when connecting to public IP of VM.

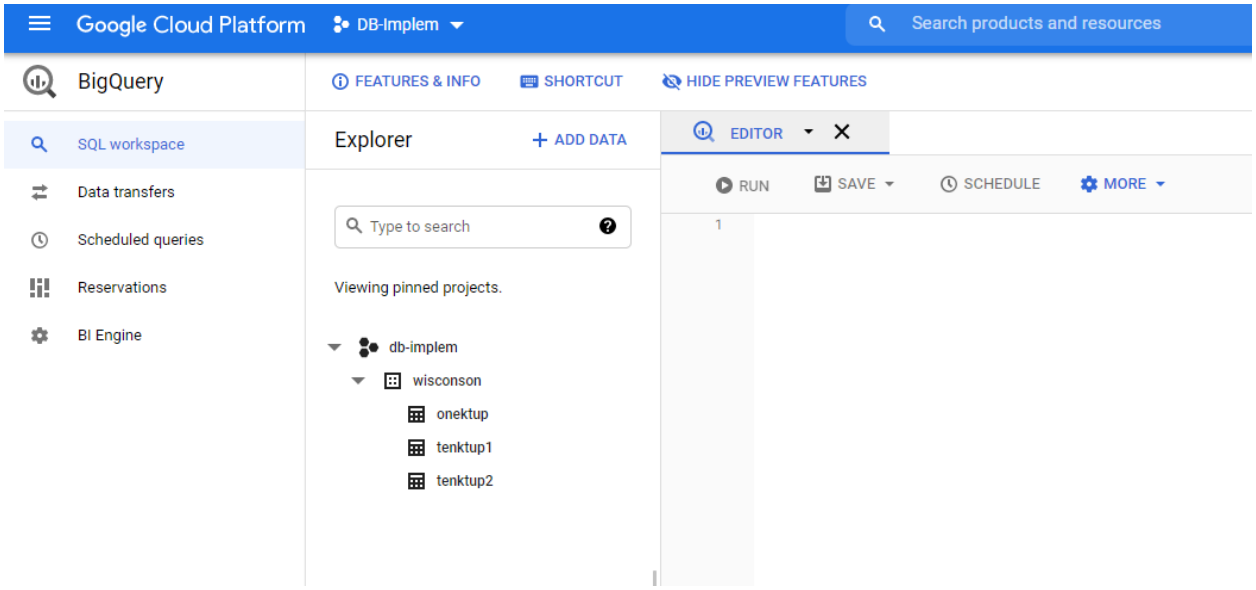
Extra credit for doing the project as a container or VM:

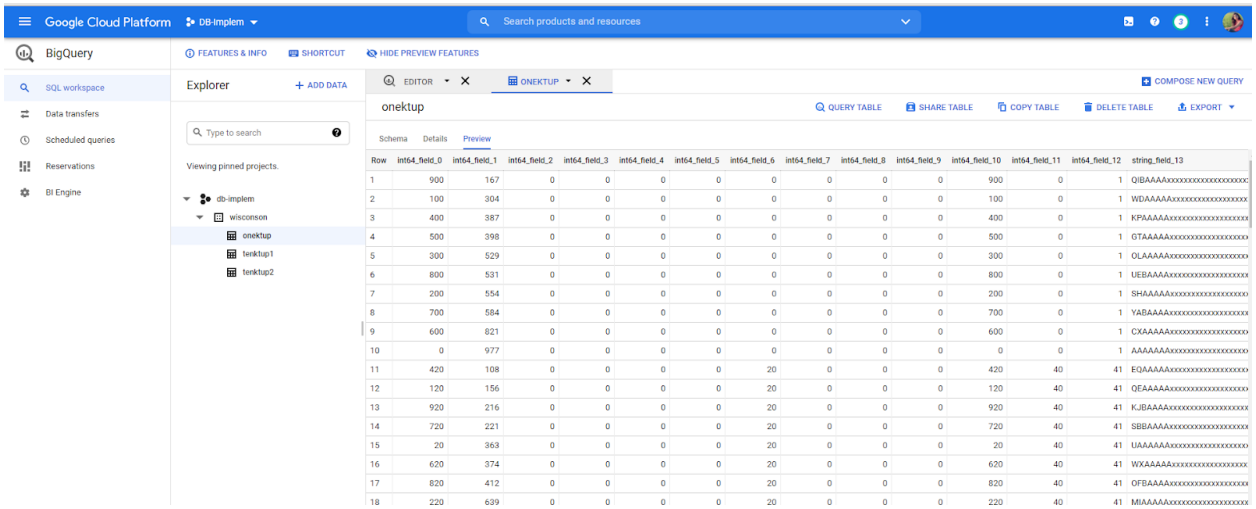
Connected postgres SQL instance to pgadmin in our local.





Loaded Data into Big Query:


Google Cloud Platform
DB-Implem
Search products and resources
BigQuery
FEATURES & INFO
SHORTCUT
HIDE PREVIEW FEATURES
SQL workspace
Data transfers
Scheduled queries
Reservations
BI Engine
Explorer
ADD DATA
EDITOR
RUN
SAVE
SCHEDULE
MORE
Type to search
Viewing pinned projects.
db-implem
wisconson
onektup
tenktup1
tenktup2
1


Google Cloud Platform
DB-Implem
Search products and resources
BigQuery
FEATURES & INFO
SHORTCUT
HIDE PREVIEW FEATURES
SQL workspace
Data transfers
Scheduled queries
Reservations
BI Engine
Explorer
ADD DATA
Type to search
Viewing pinned projects.
db-implem
wisconson
onektup
tenktup1
tenktup2
EDITOR
ONEKTUP
COMPOSE NEW QUERY
onektup
QUERY TABLE
SHARE TABLE
COPY TABLE
DELETE TABLE
EXPORT
Schema
Details
Preview
Row int64_field_0 int64_field_1 int64_field_2 int64_field_3 int64_field_4 int64_field_5 int64_field_6 int64_field_7 int64_field_8 int64_field_9 int64_field_10 int64_field_11 int64_field_12 string_field_13
1 900 167 0 0 0 0 0 0 0 0 900 0 1 QIBAAAAxxxxxxxxxxxxxxxxxxxx
2 100 304 0 0 0 0 0 0 0 0 100 0 1 WDAAAAAxxxxxxxxxxxxxxxxxxxx
3 400 387 0 0 0 0 0 0 0 0 400 0 1 KPAAAAAxxxxxxxxxxxxxxxxxxxx
4 500 398 0 0 0 0 0 0 0 0 500 0 1 GTAAAAAxxxxxxxxxxxxxxxxxxxx
5 300 529 0 0 0 0 0 0 0 0 300 0 1 OLAAAAAxxxxxxxxxxxxxxxxxxxx
6 800 531 0 0 0 0 0 0 0 0 800 0 1 UEBAAAAxxxxxxxxxxxxxxxxxxxx
7 200 554 0 0 0 0 0 0 0 0 200 0 1 SHAAAAAxxxxxxxxxxxxxxxxxxxx
8 700 584 0 0 0 0 0 0 0 0 700 0 1 YABAAAAxxxxxxxxxxxxxxxxxxxx
9 600 821 0 0 0 0 0 0 0 0 600 0 1 CXAAAAAxxxxxxxxxxxxxxxxxxxx
10 0 977 0 0 0 0 0 0 0 0 0 0 1 AAAAAAAxxxxxxxxxxxxxxxxxxxx
11 420 108 0 0 0 0 20 0 0 0 420 40 41 EQAAAAAxxxxxxxxxxxxxxxxxxxx
12 120 156 0 0 0 0 20 0 0 0 120 40 41 QEAAAAAxxxxxxxxxxxxxxxxxxxx
13 920 216 0 0 0 0 20 0 0 0 920 40 41 KJBAAAAxxxxxxxxxxxxxxxxxxxx
14 720 221 0 0 0 0 20 0 0 0 720 40 41 SBBAAAAxxxxxxxxxxxxxxxxxxxx
15 20 363 0 0 0 0 20 0 0 0 20 40 41 UAAAAAAxxxxxxxxxxxxxxxxxxxx
16 620 374 0 0 0 0 20 0 0 0 620 40 41 WXAAAAAxxxxxxxxxxxxxxxxxxxx
17 820 412 0 0 0 0 20 0 0 0 820 40 41 OFBAAAAxxxxxxxxxxxxxxxxxxxx
18 220 639 0 0 0 0 20 0 0 0 220 40 41 MIAAAAAxxxxxxxxxxxxxxxxxxxx