PG_DWH TASK 3

Query Query History

```sql
DROP TABLE IF EXISTS labs.person;

SET search_path TO labs;

CREATE TABLE labs.person (
id integer NOT NULL,
name varchar(15)
);
INSERT INTO person VALUES(1, 'Bob');
INSERT INTO person VALUES(2, 'Alice');
INSERT INTO person VALUES(3, 'Robert');
```

Data Output Messages Notifications

```
NOTICE:  table "person" does not exist, skipping
INSERT 0 1

Query returned successfully in 214 msec.
```

Table created , values are inserted.

```
SET search_path TO labs;

CREATE TABLE labs.person (
id integer NOT NULL,
name varchar(15)
);
INSERT INTO person VALUES(1, 'Bob');
INSERT INTO person VALUES(2, 'Alice');
INSERT INTO person VALUES(3, 'Robert');


CREATE TABLE IF NOT EXISTS labs.test_simple(a int,b int);

insert into test_simple values (generate_series(1,1000000));
```

Data Output | Messages | Notifications

```
INSERT 0 1000000

Query returned successfully in 2 secs 353 msec.
```

Total rows: 3 of 3 | Query complete 00:00:02.353

Simple table created and populated in 2 seconds in first query.

```
CREATE TABLE labs.person (
id integer NOT NULL,
name varchar(15)
);
INSERT INTO person VALUES(1, 'Bob');
INSERT INTO person VALUES(2, 'Alice');
INSERT INTO person VALUES(3, 'Robert');


CREATE TABLE IF NOT EXISTS labs.test_simple(a int,b int);

insert into test_simple values (generate_series(1,1000000));


insert into test_simple values (generate_series(1,5000000));
```

Data Output | Messages | Notifications

```
INSERT 0 5000000

Query returned successfully in 11 secs 619 msec.
```

Total rows: 3 of 3 | Query complete 00:00:11.619

Second query took 11 seconds to execute.

```
--INSERT INTO person VALUES(1, 'Bob');
--INSERT INTO person VALUES(2, 'Alice');
--INSERT INTO person VALUES(3, 'Robert');


CREATE TABLE IF NOT EXISTS labs.test_simple(a int,b int);

--insert into test_simple values (generate_series(1,1000000));
--insert into test_simple values (generate_series(1,5000000));


CREATE UNLOGGED TABLE IF NOT EXISTS labs.test_unlogged(a int, b int);
insert into test_unlogged values (generate_series(1,1000000));
```

Data Output | Messages | Notifications

```
INSERT 0 1000000

Query returned successfully in 1 secs 316 msec.
```

Total rows: 3 of 3 | Query complete 00:00:01.316

Unlogged table took 1.3 second to be populated 1.000.000 rows.

```
--INSERT INTO person VALUES(1, 'Bob');
--INSERT INTO person VALUES(2, 'Alice');
--INSERT INTO person VALUES(3, 'Robert');


CREATE TABLE IF NOT EXISTS labs.test_simple(a int,b int);

--insert into test_simple values (generate_series(1,1000000));
--insert into test_simple values (generate_series(1,5000000));


CREATE UNLOGGED TABLE IF NOT EXISTS labs.test_unlogged(a int, b int);
insert into test_unlogged values (generate_series(1,1000000));
insert into test_simple values (generate_series(1,5000000));
```

Data Output | Messages | Notifications

```
INSERT 0 5000000

Query returned successfully in 11 secs 114 msec.
```

Total rows: 3 of 3 | Query complete 00:00:11.114

And second one took 11 second , as we see unlogged tables are faster than usual tables.

```
CREATE TABLE labs.users(user_id INT GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY, login VARCHAR(30)) INHERITS (person);
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 154 msec.
```

Table created .

```
CREATE TABLE labs.users(user_id INT GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY, login VARCHAR(30)) INHERITS (person);

INSERT INTO users VALUES (1, 'new Bob', 'NewLogin');
INSERT INTO users VALUES (999, 'TestUser', 'TestLogin');
```

Data Output | Messages | Notifications

```
ERROR:  invalid input syntax for type integer: "NewLogin"
LINE 2: INSERT INTO users VALUES (1, 'new Bob', 'NewLogin');
                                                ^

SQL state: 22P02
Character: 42
```

Third value need to be an integer as its inherits from person.

```
CREATE TABLE labs.users(user_id INT GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY, login VARCHAR(30)) INHERITS (person);

INSERT INTO users VALUES (5, 'new Bob', 123,'loged_in');
INSERT INTO users VALUES (999, 'TestUser', 'TestLogin');
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 123 msec.
```

Inserted new row , so 5 and ‘new Bob’ values are going to be values for person table also , so it’s insering there as well.

Data Output Messages Notifications

	id integer	name character varying (15)	user_id [PK] integer	login character varying (30)
1	1	new Bob	1	[null]
2	5	new Bob	123	loged_in
3	1	new Bob	454	[null]
4	4	new Bob	666	[null]

Users table after inserting and person table .

	id integer	name character varying (15)
1	1	Bob
2	2	Alice
3	3	Robert
4	1	new Bob
5	1	new Bob
6	4	new Bob
7	5	new Bob

In other words users table will contain person table in it.

```
31
32
33
34 UPDATE person
35 SET name = 'not Bob'
36 where id = 1;
```

Data Output Messages Notifications

```
UPDATE 3

Query returned successfully in 237 msec.
```

Update statement updates all new Bobs to not Bobs in both tables .

Data Output Messages Notifications

	id integer	name character varying (15)
1	2	Alice
2	3	Robert
3	1	not Bob
4	4	new Bob
5	5	new Bob
6	1	not Bob
7	1	not Bob

Data Output Messages Notifications

	id integer	name character varying (15)	user_id [PK] integer	login character varying (30)
1	1	not Bob	1	[null]
2	5	new Bob	123	loged_in
3	1	not Bob	454	[null]
4	4	new Bob	666	[null]

Alter person.

```
ALTER TABLE person ADD COLUMN status integer DEFAULT 0;
ALTER TABLE person ADD CONSTRAINT status CHECK (status in (0,1)) NO
INHERIT;
ALTER TABLE person ADD CONSTRAINT id UNIQUE (id, name);
```

Data Output Messages Notifications

```
ALTER TABLE

Query returned successfully in 175 msec.
```

We can see that by Alterin person table users table were altered as well

Data Output Messages Notifications

	id integer	name character varying (15)	user_id [PK] integer	login character varying (30)	status integer
1	1	not Bob	1	[null]	0
2	5	new Bob	123	loged_in	0
3	1	not Bob	454	[null]	0
4	4	new Bob	666	[null]	0

2.

Table created and populated with 15776640 values.

666 MB's

```
INSERT INTO test_index(num, load_date)
SELECT random(), x
FROM generate_series('2017-01-01 0:00'::timestamptz,
 '2021-12-31 23:59:59'::timestamptz, '10 seconds'::interval) x;


SELECT pg_size_pretty(pg_relation_size('test_index'));
```

Data Output Messages Notifications

	pg_size_pretty text
1	666 MB

```
SELECT date_trunc('year', load_date), max(num)
FROM test_index
WHERE load_date BETWEEN '2021-09-01 0:00' AND '2021-10-31 11:59:59'
GROUP BY 1
ORDER BY 1;
```

Data Output | Messages | Notifications

	date_trunc timestamp with time zone	max double precision
1	2021-01-01 00:00:00+04	0.9999981809137353

Total rows: 1 of 1 | Query complete 00:00:02.612

Took 2.6seconds.

```
ORDER BY 1;

CREATE INDEX test_index_date ON test_index (load_date);
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 17 secs 484 msec.
```

Creating index took 17.4 seconds and size is 338MB.

```
CREATE INDEX test_index_date ON test_index (load_date);
SELECT pg_size_pretty(pg_relation_size('test_index_date'));
```

Data Output | Messages | Notifications

	pg_size_pretty text
1	338 MB

```
71  SELECT pg_size_pretty(pg_relation_size('test_index_date'));
72 DROP INDEX test_index_date;
73 CREATE INDEX test_index_date ON test_index (load_date);
74 CREATE INDEX test_index_load_date_brin ON test_index USING BRIN (load_date);
75
```

Data Output Messages Notifications

```
CREATE INDEX

Query returned successfully in 7 secs 682 msec.
```

BRIN index created took 7.6 sec and it's just 40kb.

```
74 CREATE INDEX test_index_load_date_brin ON test_index USING BRIN (load_date);
75  SELECT pg_size_pretty(pg_relation_size('test_index_load_date_brin'));
76
```

Data Output Messages Notifications

	pg_size_pretty text
1	40 kB

3.

Table created and populated with 10.000.000 values , size 651 MB

```
81
82 CREATE TABLE test_index AS SELECT id, md5(id::text) as t_hash
83 FROM generate_series(1, 10000000) AS id;
84
85 SELECT pg_size_pretty(pg_relation_size('test_index'));
86
```

Data Output Messages Notifications

	pg_size_pretty text
1	651 MB

```
79
80
81
82 CREATE TABLE test_index AS SELECT id, md5(id::text) as t_hash
83 FROM generate_series(1, 10000000) AS id;
84
85 SELECT pg_size_pretty(pg_relation_size('test_index'));
86 SELECT * FROM test_index WHERE t_hash LIKE '%ceea167a5a%';
```

Data Output | Messages | Notifications

	id integer	t_hash text
1	7	8f14e45fceea167a5a36dedd4bea2543

Total rows: 1 of 1 | Query complete 00:00:02.509

2.5 second to find pattern.

```
83 FROM generate_series(1, 10000000) AS id;
84
85 SELECT pg_size_pretty(pg_relation_size('test_index'));
86 SELECT * FROM test_index WHERE t_hash LIKE '%ceea167a5a%';
87
88 CREATE EXTENSION pg_trgm;
89 CREATE INDEX idx_text_index_gist ON test_index USING gist(t_hash
90 gist_trgm_ops);
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 13 min 35 secs.
```

Total rows: 1 of 1 | Query complete 00:13:35.696

Creation gist index took 13 minutes :(

```
SELECT pg_size_pretty(pg_relation_size('test_index'));
SELECT * FROM test_index WHERE t_hash LIKE '%ceea167a5a%';

CREATE EXTENSION pg_trgm;
CREATE INDEX idx_text_index_gist ON test_index USING gist(t_hash
gist_trgm_ops);


DROP INDEX idx_text_index_gist;


CREATE INDEX idx_text_index_gin ON test_index USING gin (t_hash
gin_trgm_ops);
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 4 min 31 secs.
```

Total rows: 1 of 1 | Query complete 00:04:31.598

With gin index it took 4minutes and size is 613 MB


```
SELECT pg_size_pretty(pg_relation_size('test_index'));
SELECT * FROM test_index WHERE t_hash LIKE '%ceea167a5a%';

CREATE EXTENSION pg_trgm;
CREATE INDEX idx_text_index_gist ON test_index USING gist(t_hash
gist_trgm_ops);


DROP INDEX idx_text_index_gist;


CREATE INDEX idx_text_index_gin ON test_index USING gin (t_hash
gin_trgm_ops);
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 4 min 31 secs.
```

Total rows: 1 of 1 | Query complete 00:04:31.598

```
DROP INDEX idx_text_index_gist;


CREATE INDEX idx_text_index_gin ON test_index USING gin (t_hash
gin_trgm_ops);

SELECT pg_size_pretty(pg_relation_size('idx_text_index_gin'));
DROP TABLE test_index


CREATE TABLE test_index AS SELECT id, md5(id::text) as t_hash
FROM generate_series(1, 10000000) AS id;
```

Data Output | Messages | Notifications

```
SELECT 10000000

Query returned successfully in 1 min 4 secs.
```

Total rows: 10 of 10 | Query complete 00:01:04.318

Population table this time took 1 minute which is way faster then first time.

4.

```
CREATE FOREIGN TABLE labs.test_foreign_table
(
 LatD INT,
 LatM INT,
 LatS INT,
 NS TEXT,
 LonD INT,
 LonM INT,
 LonS INT,
 EW TEXT,
 City TEXT,
 State TEXT
)
SERVER test_import
OPTIONS
(
 filename '/home/styop/Downloads/ities_list.csv',
 format 'csv',
 header 'true',
 delimiter ','
);
```

Data Output | Messages | Notifications

```
CREATE FOREIGN TABLE

Query returned successfully in 115 msec.
```

Foreign table created using csv file.

```
 City TEXT,
 State TEXT
)
SERVER test_import
OPTIONS
(
 filename '/home/styop/Downloads/cities_list.csv',
 format 'csv',
 header 'true',
 delimiter ','
);


SELECT * FROM test_foreign_table;
```

Data Output | Messages | Notifications

	latd integer	latm integer	lats integer	ns text	lond integer	lonm integer	lons integer	ew text	city text	state text
1	41	5	59	N	80	39	0	W	Youngstown	OH
2	42	52	48	N	97	23	23	W	Yankton	SD
3	46	35	59	N	120	30	36	W	Yakima	WA
4	42	16	12	N	71	48	0	W	Worcester	MA
5	43	37	48	N	89	46	11	W	WisconsinDells	WI
6	36	5	59	N	80	15	0	W	Winston-Salem	NC
7	49	52	48	N	97	9	0	W	Winnipeg	MB
8	39	11	23	N	78	9	36	W	Winchester	VA
9	34	14	24	N	77	55	11	W	Wilmington	NC
10	39	45	0	N	75	33	0	W	Wilmington	DE
11	48	9	0	N	103	37	12	W	Williston	ND

Total rows: 128 of 128 | Query complete 00:00:00.134

There is 128 rows.

```
(
 filename '/home/styop/Downloads/cities_list.csv',
 format 'csv',
 header 'true',
 delimiter ','
);


SELECT * FROM test_foreign_table;

SELECT count(*) FROM test_foreign_table;


CREATE MATERIALIZED VIEW mview as  (  SELECT * FROM test_foreign_table);
```

Data Output | Messages | Notifications

```
SELECT 128

Query returned successfully in 260 msec.
```

Total rows: 1 of 1 | Query complete 00:00:00.260

Materialized view created with 128 rows.

I deleted 2 rows from file but the count in materialized view remains same, as we store our previous data in materialized view (it means we copy the file in the view)

```
 format 'csv',
 header 'true',
 delimiter ','
);


SELECT * FROM test_foreign_table;

SELECT count(*) FROM test_foreign_table;


CREATE MATERIALIZED VIEW mview as  (  SELECT * FROM test_foreign_table);
select count(*) from mview
```

Data Output | Messages | Notifications

	count bigint
1	128

Total rows: 1 of 1 | Query complete 00:00:00.210