1.

test_db/postgres@PostgreSQL 17*

```
SELECT d.oid, d.datname, d.datistemplate, d.datallowconn, t
FROM pg_database d
JOIN pg_tablespace t ON t.oid = d.dattablespace;
```

	oid oid	datname name	datistemplate boolean	datallowconn boolean	spcname name
1	5	postgres	false	true	pg_default
2	1	template1	true	true	pg_default
3	4	template0	true	false	pg_default
4	17463	gym_membership	false	true	pg_default
5	16776	reweb_old	false	true	pg_default
6	18585	reweb	false	true	pg_default
7	20302	AuctionHouseDB	false	true	pg_default
8	20709	sales_history	false	true	pg_default
9	28810	RealEstateDB	false	true	pg_default
10	16852	dvdrental	false	true	pg_default
11	34470	VehiclesSales	false	true	pg_default
12	39126	VehilesSales2	false	true	pg_default
13	42500	ESP	false	true	pg_default
14	43559	test_db	false	true	pg_default

2.

test_db/postgres@PostgreSQL 17*

```
select *
from pg_tablespace
```

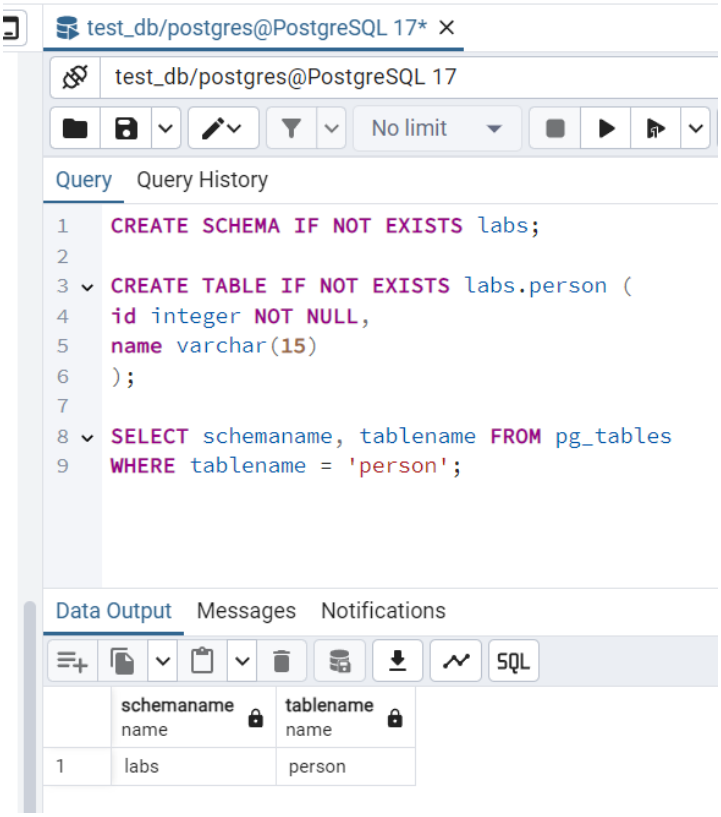
	oid [PK] oid	spcname name	spcowner oid	spcacl aclitem[]	spcoptions text[]
1	1663	pg_default	10	[null]	[null]
2	1664	pg_global	10	[null]	[null]
3	43564	mytablespace	10	[null]	[null]

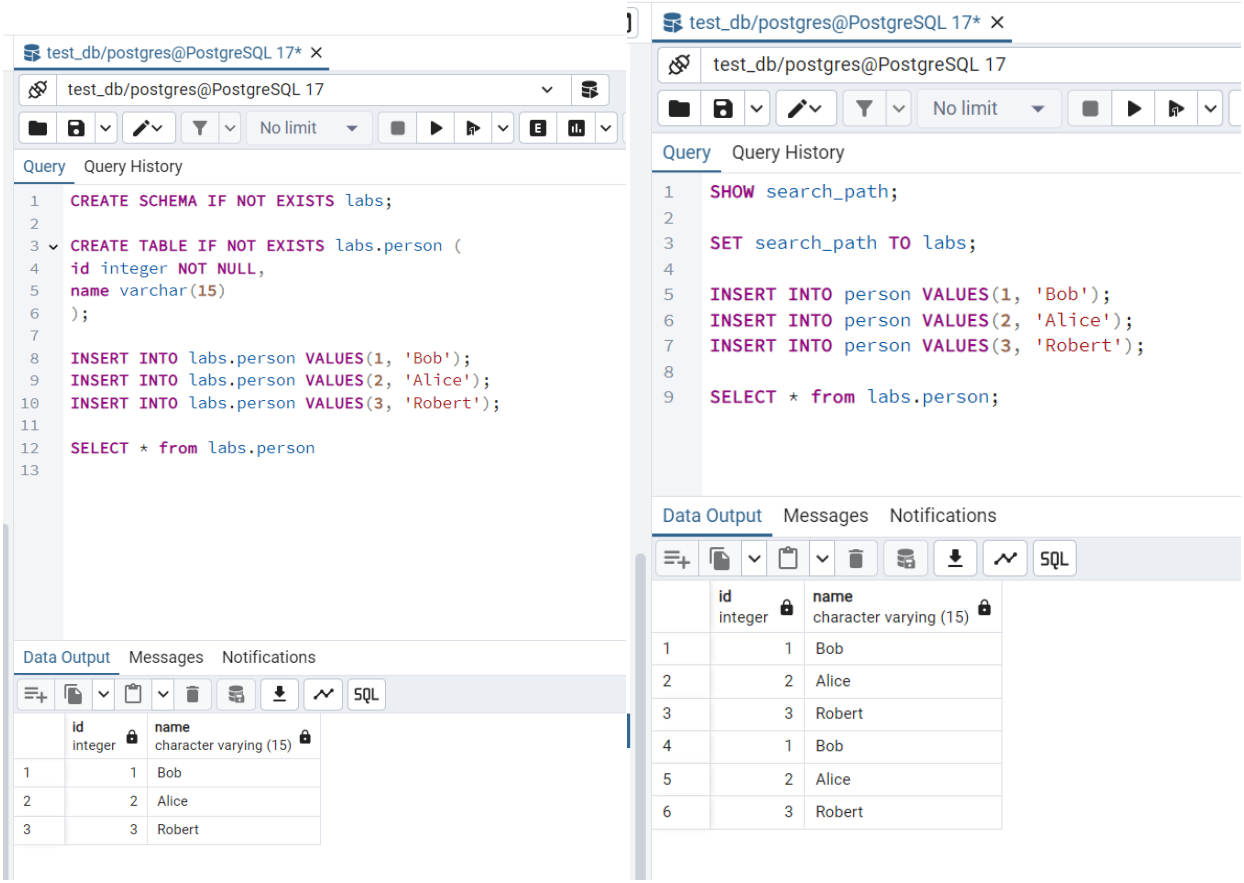
AuctionHouseDB/postgres@PostgreSQL 17*

```
select d.oid, d.datname, d.datistemplate, d.datallowconn, t
from pg_database d
join pg_tablespace t on t.oid = d.dattablespace
```

	oid oid	datname name	datistemplate boolean	datallowconn boolean	spcname name
1	5	postgres	false	true	pg_default
2	1	template1	true	true	pg_default
3	4	template0	true	false	pg_default
4	17463	gym_membership	false	true	pg_default
5	16776	reweb_old	false	true	pg_default
6	18585	reweb	false	true	pg_default
7	20302	AuctionHouseDB	false	true	pg_default
8	20709	sales_history	false	true	pg_default
9	28810	RealEstateDB	false	true	pg_default
10	16852	dvdrental	false	true	pg_default
11	34470	VehiclesSales	false	true	pg_default
12	39126	VehilesSales2	false	true	pg_default
13	42500	ESP	false	true	pg_default
14	43559	test_db	false	false	mytablespace

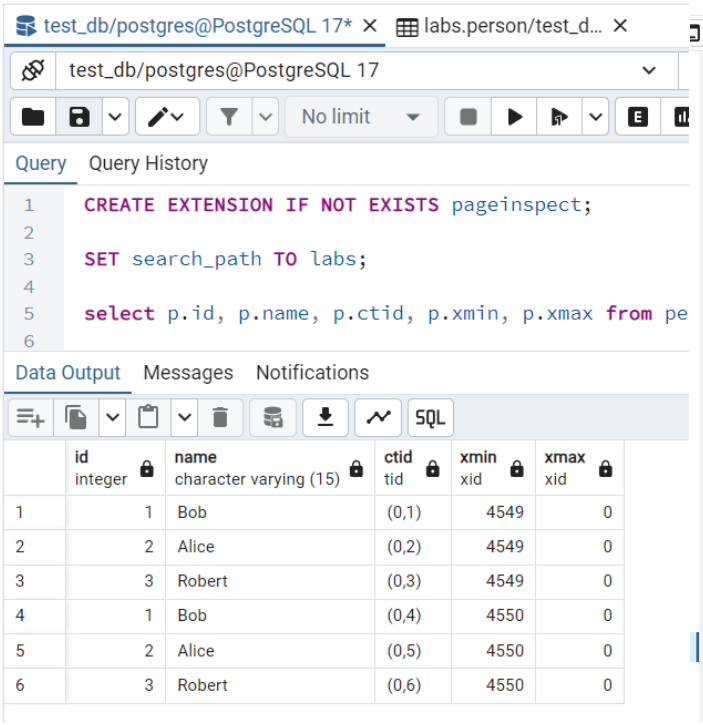
3.

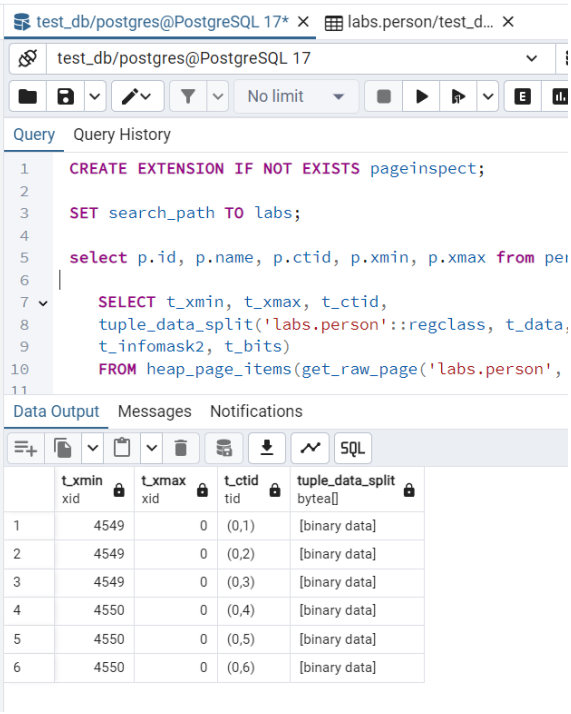


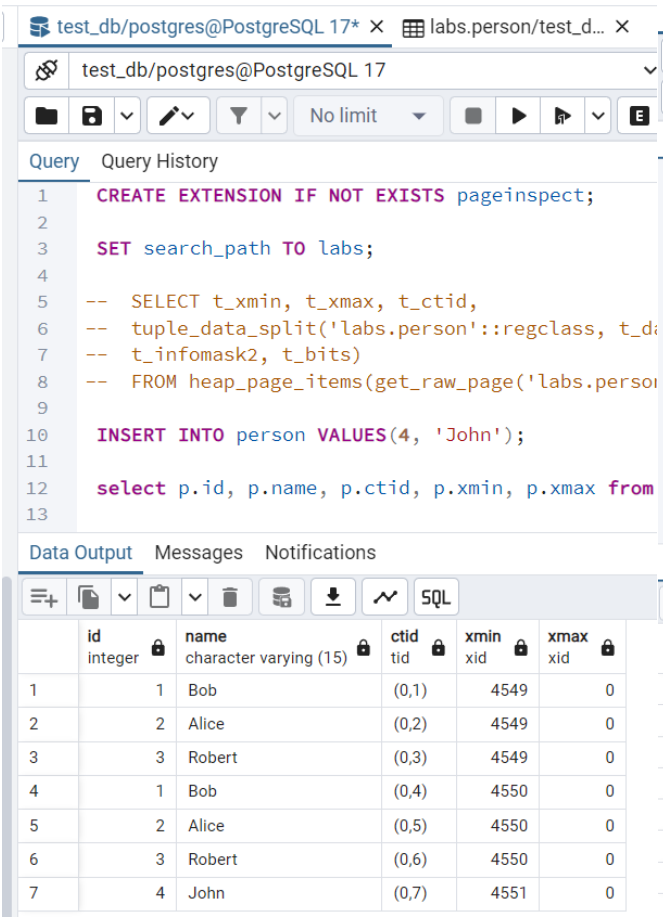


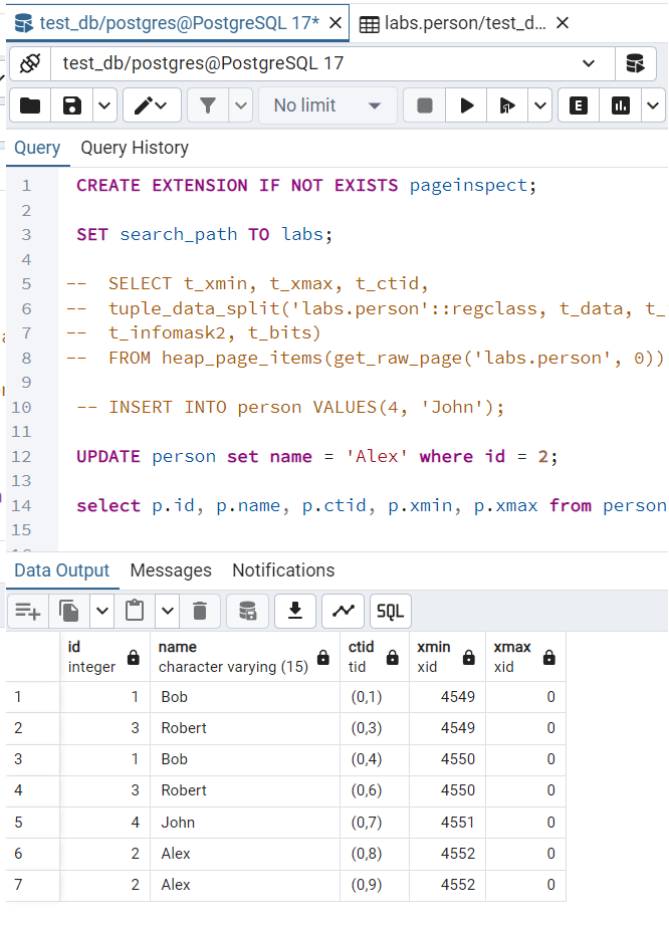
Key Takeaways: Set `search_path` allows insert records without specifying the schema name

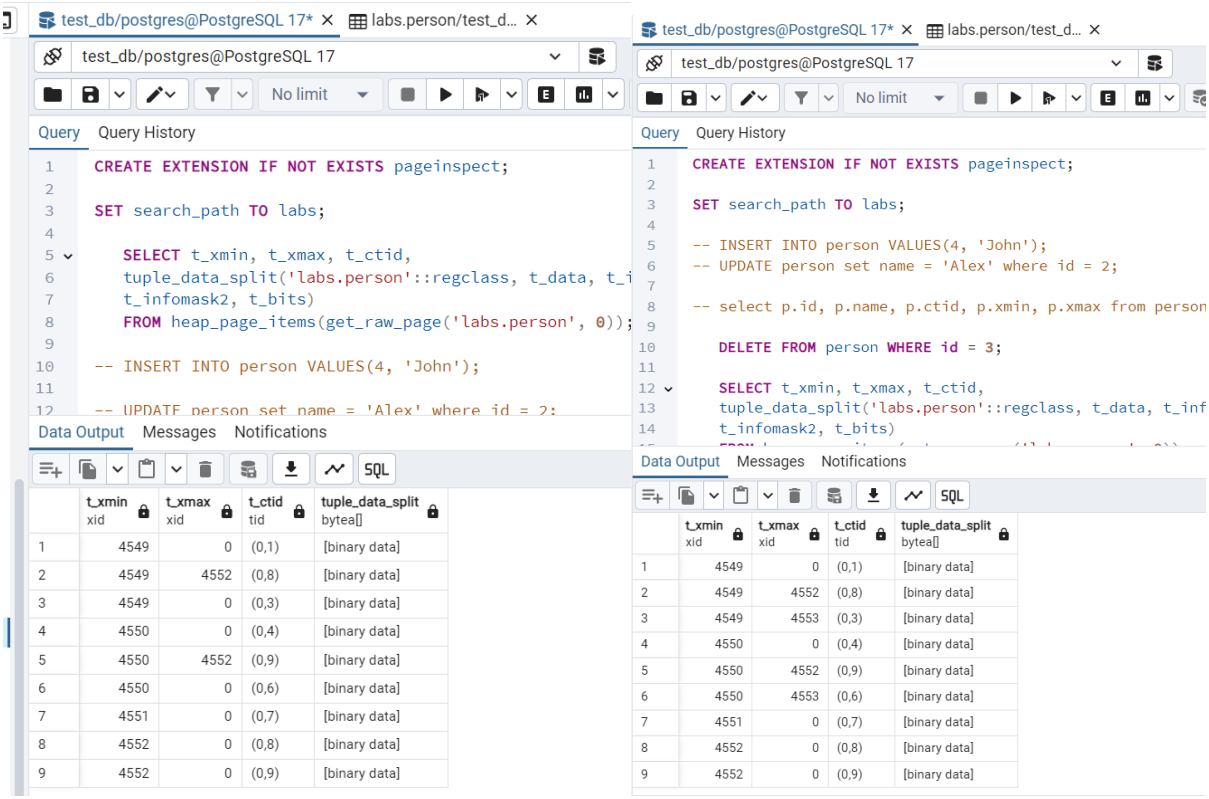
4.

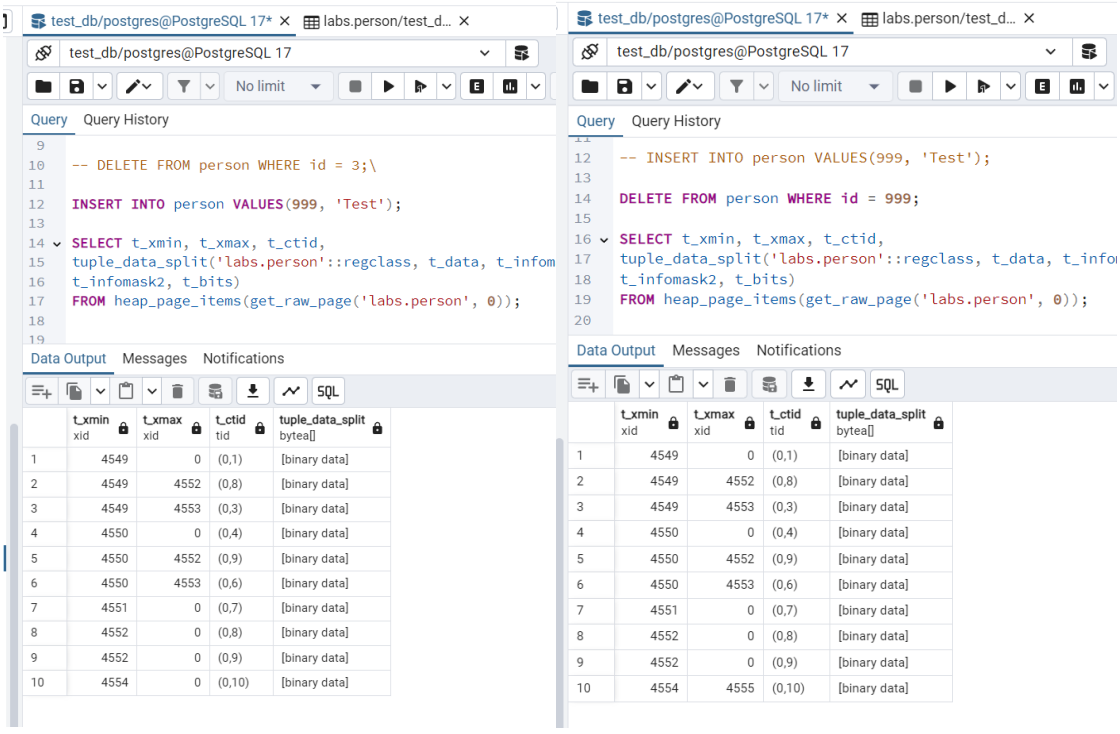

test_db/postgres@PostgreSQL 17*
labs.person/test_d...
test_db/postgres@PostgreSQL 17
No limit
Query
Query History
1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 select p.id, p.name, p.ctid, p.xmin, p.xmax from pe
6
Data Output
Messages
Notifications
SQL
id integer
name character varying (15)
ctid tid
xmin xid
xmax xid
1 1 Bob (0,1) 4549 0
2 2 Alice (0,2) 4549 0
3 3 Robert (0,3) 4549 0
4 1 Bob (0,4) 4550 0
5 2 Alice (0,5) 4550 0
6 3 Robert (0,6) 4550 0


test_db/postgres@PostgreSQL 17*
labs.person/test_d...
test_db/postgres@PostgreSQL 17
No limit
Query
Query History
1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 select p.id, p.name, p.ctid, p.xmin, p.xmax from pe
6
7 SELECT t_xmin, t_xmax, t_ctid,
8 tuple_data_split('labs.person'::regclass, t_data,
9 t_infomask2, t_bits)
10 FROM heap_page_items(get_raw_page('labs.person',
11
Data Output
Messages
Notifications
SQL
t_xmin xid
t_xmax xid
t_ctid tid
tuple_data_split bytea[]
1 4549 0 (0,1) [binary data]
2 4549 0 (0,2) [binary data]
3 4549 0 (0,3) [binary data]
4 4550 0 (0,4) [binary data]
5 4550 0 (0,5) [binary data]
6 4550 0 (0,6) [binary data]


test_db/postgres@PostgreSQL 17*
labs.person/test_d...
test_db/postgres@PostgreSQL 17
No limit
Query
Query History
1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 -- SELECT t_xmin, t_xmax, t_ctid,
6 -- tuple_data_split('labs.person'::regclass, t_da
7 -- t_infomask2, t_bits)
8 -- FROM heap_page_items(get_raw_page('labs.perso
9
10 INSERT INTO person VALUES(4, 'John');
11
12 select p.id, p.name, p.ctid, p.xmin, p.xmax from
13
Data Output
Messages
Notifications
SQL
id integer
name character varying (15)
ctid tid
xmin xid
xmax xid
1 1 Bob (0,1) 4549 0
2 2 Alice (0,2) 4549 0
3 3 Robert (0,3) 4549 0
4 1 Bob (0,4) 4550 0
5 2 Alice (0,5) 4550 0
6 3 Robert (0,6) 4550 0
7 4 John (0,7) 4551 0


test_db/postgres@PostgreSQL 17*
labs.person/test_d...
test_db/postgres@PostgreSQL 17
No limit
Query
Query History
1 CREATE EXTENSION IF NOT EXISTS pageinspect;
2
3 SET search_path TO labs;
4
5 -- SELECT t_xmin, t_xmax, t_ctid,
6 -- tuple_data_split('labs.person'::regclass, t_data, t_
7 -- t_infomask2, t_bits)
8 -- FROM heap_page_items(get_raw_page('labs.person', 0))
9
10 -- INSERT INTO person VALUES(4, 'John');
11
12 UPDATE person set name = 'Alex' where id = 2;
13
14 select p.id, p.name, p.ctid, p.xmin, p.xmax from person
15
Data Output
Messages
Notifications
SQL
id integer
name character varying (15)
ctid tid
xmin xid
xmax xid
1 1 Bob (0,1) 4549 0
2 3 Robert (0,3) 4549 0
3 1 Bob (0,4) 4550 0
4 3 Robert (0,6) 4550 0
5 4 John (0,7) 4551 0
6 2 Alex (0,8) 4552 0
7 2 Alex (0,9) 4552 0

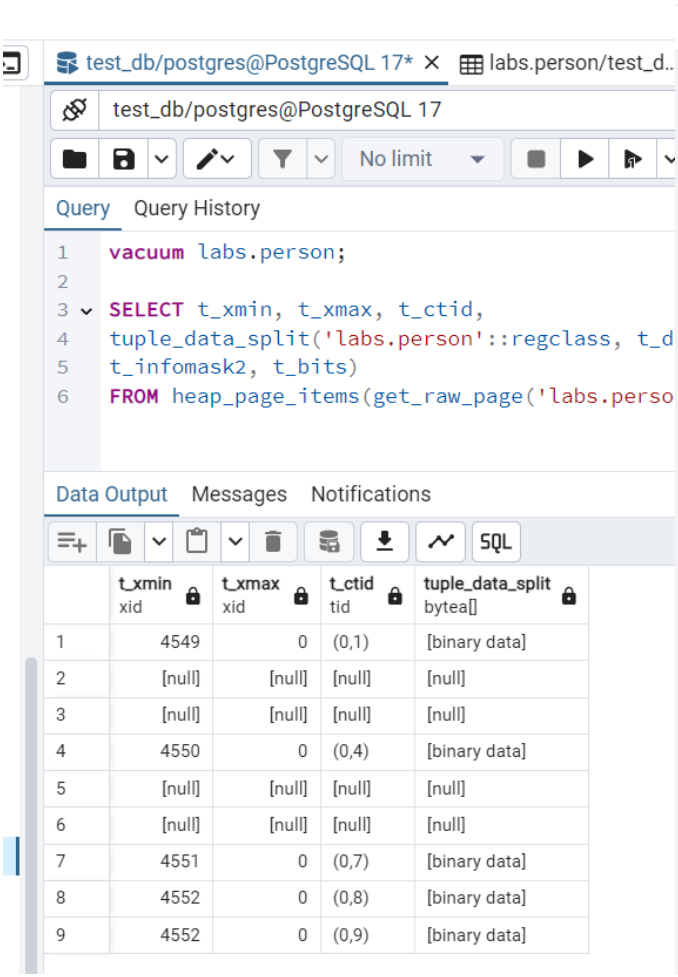


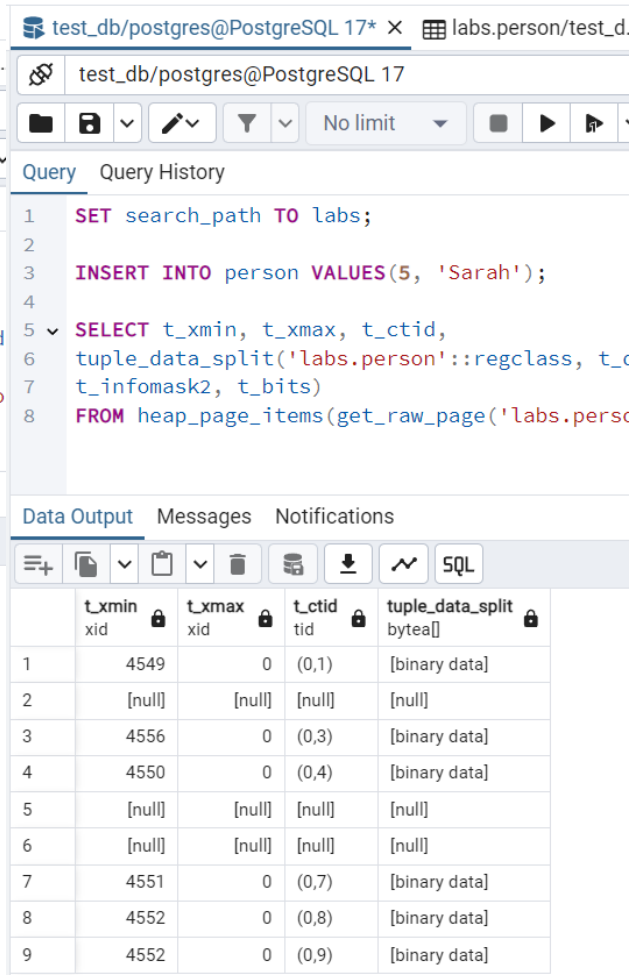


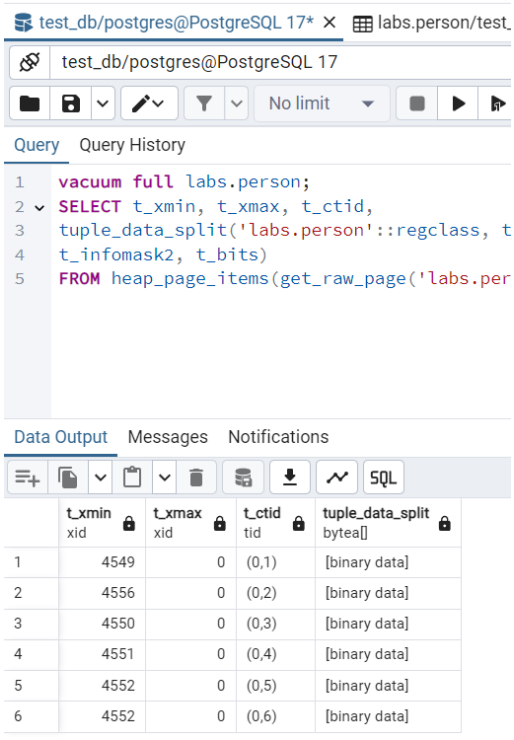
Key Takeaways:

- `pageinspect` shows internal row versioning, helping to understand MVCC behavior.
- PostgreSQL does not update rows directly. Instead, it creates a new row version (UPDATE behaves like an INSERT + DELETE).
- Deleted rows remain in the database but are marked as obsolete (xmax is assigned).
- New inserts always get a fresh xmin, representing their transaction ID.

5.







Key Takeaways:

- Before VACUUM: Deleted and updated rows exist as dead tuples.
- After VACUUM: Dead tuples are marked as reusable space.
- After VACUUM FULL: Dead tuples are completely removed, reducing table size.
- Inserting new rows after VACUUM: PostgreSQL reuses free space instead of expanding the table file.