BUIT IN FUNCTIONS IN SQL:

- String Functions
- Math/ Numeric Functions

String BUILT-IN FUNCTIONS:

SELECT ASCII('A');



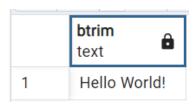
SELECT CONCAT('Hello','World!');



SELECT LOWER('PERIYAKARUPPAN');



SELECT TRIM(' Hello World! ');



SELECT REVERSE('ZOHO');



MATH/NUMERIC BUILT-IN FUNCTIONS:

SELECT ABS(-255.43);



SELECT ROUND(255.43);

	round numeric
1	255

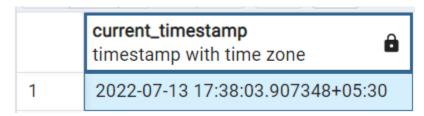
SELECT POWER(5,3);



SELECT SQRT(25);



SELECT CURRENT_TIMESTAMP;



TRIGGERS:

I have created a sample table **records** consisting of id, name and address.

I have used a **trigger** function that gets triggered once I update the address in records table.

	id integer	name character varying (50)	address character varying (255)
1	1	Peri	Trichy
2	2	Yash	Madurai
3	3	Mahesh	Chennai
4	4	Arun	Mumbai

Trigger: Trigger is a function that executes automatically if it satisfies a particular function. Then it executes the user defined function.

Code of Creating a Procedure/Function:

create or replace function add_func()
returns trigger as
\$BODY\$
begin
if NEW.address <> OLD.address then
insert into address_log values (old.id,old.name,old.address,new.address);
end if;
return new;
end;
\$BODY\$

language plpgsql;

Code for Creating a trigger and mapping it to execute the required function:

create trigger trigger_func

before update

on records

for each row

execute procedure add_func();

So if I update address of a person in row, old and new address gets stored into new row of the address_log table.

update records set address='Bangalore' where id=1;

select * from records;

	id integer	name character varying (50)	address character varying (255)
1	2	Yash	Madurai
2	3	Mahesh	Chennai
3	4	Arun	Mumbai
4	1	Peri	Bangalore

Select * from address_log;

		id integer	name character varying (50)	old_address character varying (255)	new_address character varying (255) •	
ı	1	1	Peri	Trichy	Bangalore	

TRANSACTIONS:

Transactions in SQL follow ACID properties.

A – Atomicity

C – Consistency

I – Isolation

D - Durability

The most important commands in a transaction are

- COMMIT
- ROLLBACK
- SAVEPOINT
- SET TRANSACTION

COMMIT:

I will start the transaction by

BEGIN;

BEGIN Query returned successfully in 54 msec.

I have created a movie DB with sample data.

Select * from movies;

	name character varying (50)	actor character varying (20)	actress character varying (20)	director character varying (20)	year_of_release integer
1	Spider-Man No Way Ho	Tom Holland	Zendaya	Jon Watts	2021
2	Master	Vijay	Malavika Mohan	Lokesh Kanagaraj	2021
3	Premam	Nivin Pauli	Sai Pallavi	Alphonse Puthren	2015
4	Baahubali	Prabhas	Anushka	Rajamouli	2015
5	Avatar	Sam Worthington	Zoe Saldana	James Cameron	2009
6	RRR	Ram Charan, NTR	Alia Bhatt	Rajamouli	2021

I'm Performing a delete operation and saving it with the COMMIT;

delete from movies where name = 'RRR';

COMMIT;

Output

COMMIT

Query returned successfully in 698 msec.

The resultant deleted table after commit is:

Select * from movies;

	name character varying (50)	actor character varying (20)	actress character varying (20)	director character varying (20)	year_of_release integer
1	Spider-Man No Way H	Tom Holland	Zendaya	Jon Watts	2021
2	Master	Vijay	Malavika Mohan	Lokesh Kanagaraj	2021
3	Premam	Nivin Pauli	Sai Pallavi	Alphonse Puthren	2015
4	Baahubali	Prabhas	Anushka	Rajamouli	2015
5	Avatar	Sam Worthington	Zoe Saldana	James Cameron	2009

Now I want to rollback to previous savepoint that is the begin point so the delete operation will be un done.

ROLLBACK;

	name character varying (50)	actor character varying (20)	actress character varying (20)	director character varying (20)	year_of_release integer
1	Spider-Man No Way Ho	Tom Holland	Zendaya	Jon Watts	2021
2	Master	Vijay	Malavika Mohan	Lokesh Kanagaraj	2021
3	Premam	Nivin Pauli	Sai Pallavi	Alphonse Puthren	2015
4	Baahubali	Prabhas	Anushka	Rajamouli	2015
5	Avatar	Sam Worthington	Zoe Saldana	James Cameron	2009
6	RRR	Ram Charan, NTR	Alia Bhatt	Rajamouli	2021

SAVEPOINT:

BEGIN SAVEPOINT S1;

SAVEPOINT Query returned successfully in 52 msec.

DELETE FROM movies where name = 'Master';

select * from movies;

	name character varying (50)	actor character varying (20)	actress character varying (20)	director character varying (20)	year_of_release integer
1	Spider-Man No Way Ho	Tom Holland	Zendaya	Jon Watts	2021
2	Premam	Nivin Pauli	Sai Pallavi	Alphonse Puthren	2015
3	Baahubali	Prabhas	Anushka	Rajamouli	2015
4	Avatar	Sam Worthington	Zoe Saldana	James Cameron	2009

ROLLBACK TO S1;

ROLLBACK Query returned successfully in 66 msec.

SELECT * From movies;

	name character varying (50)	actor character varying (20)	actress character varying (20)	director character varying (20)	year_of_release integer
1	Spider-Man No Way Ho	Tom Holland	Zendaya	Jon Watts	2021
2	Master	Vijay	Malavika Mohan	Lokesh Kanagaraj	2021
3	Premam	Nivin Pauli	Sai Pallavi	Alphonse Puthren	2015
4	Baahubali	Prabhas	Anushka	Rajamouli	2015
5	Avatar	Sam Worthington	Zoe Saldana	James Cameron	2009

MASTER Movie deletion is UNDONE my ROLLBACK;

SET TRANSACTION:

It will set read / read write mode for a transaction

SET TRANSACTION READ ONLY;

DELETE FROM movies where name = 'Master';

ERROR: cannot execute DELETE in a read-only transaction SQL state: 25006