Functions:

1

```
create function increase_year(pname varchar(30))
	returns void as $$
begin
	if(select p.years_served from president p where p."name" = pname) > 8 then
		raise exception 'Years are already 8';
	else update president
	set years_served = years_served + 1
	where "name" = pname;
end if;
end; $$ language plpgsql;
```

2

```
create function add_hobby(pid int, nhobby varchar(30))
	returns void as $$
begin
	if(select count(p.hobby) from pres_hobby p where p.hobby = nhobby) = 0 then
	raise exception 'Hobby does not exist';
	else
		insert into pres_hobby(pres_id, hobby) values (pid, nhobby);
	end if;
end; $$ language plpgsql;
```

3

```
create function add_tenure(pid int, adminnr int, presid int, thisyear int)
	returns void as $$
begin
	if(select count(a.admin_nr) from administration a where a.admin_nr + 1 =
adminnr) = 1 then
	insert into administration (id, admin_nr, pres_id, year_inaugurated)
	values (pid, adminnr, presid, thisyear);
	else raise exception 'Admin nr already exists';
end if;
end; $$ language plpgsql;
```

4

```
create function add_child(pid int, sname varchar(30), uchildren int)
	returns void as $$
begin
	if(select p.nr_children from pres_marriage p where p.pres_id = pid) +
uchildred <= p.nr_children then
	raise exception 'Cannot work';
	else update pres_marriage pm
	set pm.nr_children = pm.nr_children + uchildren
	where pm.pres_id = pid;
end if;
end; $$ language plpgsql;
```

Cursor

1

```
CREATE OR replace FUNCTION viceAndPres()
	RETURNS TABLE (v_pres_name varchar(255)) AS $$
DECLARE
	v_pres CURSOR FOR
		SELECT distinct vice_pres_name
			FROM admin_vpres
			order by vice_pres_name;
	pres admin_vpres.vice_pres_name%type;
BEGIN
	create temp table if not exists temp_table (
		name varchar(255)
	);
	open v_pres;
	fetch v_pres into pres;

	while found loop
		if pres in (select distinct name from president order by name) then
			insert into temp_table
				values (pres);

end if;
		fetch v_pres into pres;
	end loop;
	close v_pres;

	return QUERY
	select name
	from temp_table;

	DROP TABLE temp_table;
END; $$ language plpgsql;

select viceAndPres();
```

2

```
create type unm_ret as (pres_name varchar(20), birth_year int4);

create or replace function unmarriedPres()
	returns setof unm_ret as $$
declare
	m_pres cursor for
		select distinct id, name, p.birth_year
			from president p
			order by id;
	p_id president.id%type;
	p_name president.name%type;
	p_birth president.birth_year%type;
begin
	create temp table if not exists notMarried_temp (
		presname varchar(20),
		birth_yr int4
	);

	OPEN m_pres;
	FETCH m_pres INTO p_id, p_name, p_birth;
```

```
	while FOUND loop
		if not p_id IN (SELECT DISTINCT pres_id FROM pres_marriage pm) THEN
			INSERT INTO notMarried_temp
				VALUES (p_name, p_birth);
		END IF;
		FETCH m_pres INTO p_id, p_name, p_birth;
	END loop;
	CLOSE m_pres;

	RETURN QUERY
		SELECT presname, birth_yr
		FROM notMarried_temp;

	drop table notMarried_temp;
end; $$ language plpgsql;

select unmarriedPres();
```

3

```
create type chi_ret as (pres_name varchar(20), children int4);

create or replace function presWithChildren()
	returns setof chi_ret as $$
declare
	p_pres cursor for
		select id, name
			from president
			order by id;
	p_name president.name%type;
	p_id president.id%type;
begin
	create temp table if not exists presAndChildren_temp (
		presname varchar(20),
		nr_children int4
	);

	open p_pres;
	fetch p_pres into p_id, p_name;

	while found loop
		if not p_id in (select distinct pres_id from pres_marriage) then
			insert into presAndChildren_temp
				values (p_name, 0);
		else
			insert into presAndChildren_temp
				values (p_name, (select sum(nr_children) from
pres_marriage where pres_id = p_id group by p_id));
		end if;
		fetch p_pres into p_id, p_name;
	end loop;
	close p_pres;

	return query
		select *
		from presAndChildren_temp;

	drop table presAndChildren_temp;
end; $$ language plpgsql;

select presWithChildren();
```

4

```
create type el_ret as (pres_name varchar(20), el_count int4);

create or replace function presElections()
	returns setof el_ret as $$
declare
	p_pres cursor for
		select name
			from president
			order by id;
	p_name president.name%type;
begin
	create temp table if not exists presAndelection_temp (
		presname varchar(20),
		elections int4
	);

	open p_pres;
	fetch p_pres into p_name;

	while found loop
		insert into presAndelection_temp
			values (p_name, (select count(*) from election where candidate
= p_name));
		fetch p_pres into p_name;
	end loop;
	close p_pres;

	return query
		select *
		from presAndelection_temp;

	drop table presAndelection_temp;
end; $$ language plpgsql;

select presElections();
```

Trigger

1

```
create function checkwin()
	returns trigger as $$
begin
	if (select count(e.winner_loser_indic) from election e where
winner_loser_indic = 'W' and election_year = new.election_year) > 0 then
		raise exception 'Number of winners invalid';
	end if;
	return new;
end; $$ language plpgsql;

create trigger checkwin
	before insert or update on election
	for each row execute procedure checkwin();
```

2

?

3

```
create function checkHobby()
	returns trigger as $$
begin
	if 'TOUCH FOOTBALL' in (select ph.hobby from pres_hobby ph inner join
president p on ph.pres_id = p.id where p.birth_year < '1800') then
		raise exception 'No touch football in those times';
	end if;
	return new;
end; $$ language plpgsql;

create trigger chechHobby
	before insert or update on pres_hobby
	for each row execute procedure checkHobby();
```

4

```
create function checkPres()
	returns trigger as $$
begin
	if((select count(a.pres_id) from administration a inner join president p on
a.pres_id = p.id where a.year_inaugurated <= p.birth_year + 21) < (select
count(pres_id) from administration)) then
		raise exception 'President too young';
	end if;
	return new;
end; $$ language plpgsql;

create trigger checkPres
	before insert or update on administration
	for each row execute procedure checkPres();
```

5

```
create function checkYearsServed()
	returns trigger as $$
begin
	if(new.years_served < old.years_served) then
		update president
		set years_served = old.years_served;
		raise notice 'Cannot decrease years served';
	end if;
	return new;
end; $$ language plpgsql;

create trigger checkYearsServed
	after update of years_served on president
	for each row execute procedure checkYearsServed();
```