IT214
DATABASE MANAGEMENT SYSTEM

PROJECT SUBMISSION

Title: - Municipal Corporation

Stored Procedures and Triggers

Team Details

NAME	ID
Karpit Patel	201701174
Dharmin Solanki	201701198
Ruchit Solanki	201701199
Jaymin Parmar	201701203

Stored Procedures: -

1. **This Stored Procedures is for calculate the tax imposed on property by municipal corporation and addition with service charge property uses and penalty if any.**

```
create or replace function compute_tax() returns void as $body$
declare
        pro1 property%rowtype;
        pro2 propertyusesServices%rowtype;
        ser  services%rowtype;
        val float;
begin

        for pro1 in select * from property loop
                val:= pro1.tax;
                if pro1.type = 'House' then
                        val = val + pro1.span*1.6;
                else
                        val = val + pro1.span*3.0;
                end if;

                for pro2 in select * from propertyusesServices loop
                        if pro1.proid = pro2.proid then
                                for ser in select * from services loop
                                        if pro2.serviceid = ser.serviceid then
                                                val = val +
pro2.units*ser.service_charge;
                                        end if;
                                end loop;
                        end if;
                end loop;
                update property set tax = val  where pro1.proid=proid;
        end loop;

end
$body$ language 'plpgsql';
select * from compute_tax();
```

2. **This Stored Procedures is for add tuple in penalty table. This table is collection of penalty which is important.**

```
create or replace function compute_penalty() returns void as $body$
declare
        pro1 property%rowtype;
        pen penalty%rowtype;
        val float;
        size int;
begin
        for pro1 in select * from property loop
                val:= pro1.tax;
                if pro1.paidstatus = 'Unpaid' then
                        val = val + val*0.1;
                        BEGIN
                                size = 0;
                                for pen in select * from penalty loop
                                        size = size + 1;
                                end loop;
                                insert into penalty (penaltyid,amount,propertyid)
                                values(size+1,val,pro1.proid);
                        END;
                end if;
                update property set tax = val where pro1.proid=proid;
        end loop;
end
$body$ language 'plpgsql';
select * from compute_penalty();
```

Triggers: -

1. **Trigger to add the cost on resource in expenses of respective year.**

```
CREATE OR REPLACE FUNCTION cost_of_resources()
RETURNS TRIGGER AS $CostonResource$
BEGIN
        IF TG_OP = 'INSERT' THEN
                UPDATE expenses SET amount=amount+NEW.cost WHERE
year=NEW.year;
                RETURN NEW;
        ELSIF TG_OP='UPDATE' THEN
                UPDATE expenses SET amount=amount+NEW.cost-OLD.cost WHERE
year=NEW.year;
                RETURN NEW;
        ELSIF TG_OP='DELETE' THEN
                UPDATE expenses SET amount=amount-OLD.cost WHERE
year=OLD.year;
                RETURN NEW;
        END IF;
END;
$CostonResource$ LANGUAGE 'plpgsql';

CREATE TRIGGER Expenses AFTER INSERT OR UPDATE OR DELETE ON
CostonResource
FOR EACH ROW EXECUTE PROCEDURE cost_of_resources();
```

2. **Trigger to add the cost on construction in expenses of respective year.**

```
CREATE OR REPLACE FUNCTION cost_of_construction()
RETURNS TRIGGER AS $CostonConstruction$
BEGIN
        IF TG_OP = 'INSERT' THEN
                UPDATE expenses SET amount=amount+NEW.cost WHERE
year=NEW.year;
                RETURN NEW;
        ELSIF TG_OP='UPDATE' THEN
                UPDATE expenses SET amount=amount+NEW.cost-OLD.cost
WHERE year=NEW.year;
                RETURN NEW;
        ELSIF TG_OP='DELETE' THEN
                UPDATE expenses SET amount=amount-OLD.cost WHERE
year=OLD.year;
                RETURN NEW;
        END IF;
END;
$CostonConstruction$ LANGUAGE 'plpgsql';

CREATE TRIGGER Expenses AFTER INSERT OR UPDATE OR DELETE ON
```

```
costOnConstruction
FOR EACH ROW EXECUTE PROCEDURE cost_of_construction();
```

3. Trigger of effect of payment on tax e.g., change the paid status to ‘unpaid’ to ‘paid’.

```
CREATE OR REPLACE FUNCTION payment_Ef()
RETURNS TRIGGER AS $paymentef$
BEGIN
        UPDATE property SET tax='0' WHERE proid=NEW.property;
        UPDATE property SET        paidstatus='paid' WHERE
proid=NEW.property;
        RETURN NEW;
END;
$paymentef$ LANGUAGE 'plpgsql';

CREATE TRIGGER property AFTER INSERT ON payment
FOR EACH ROW EXECUTE PROCEDURE payment_Ef();
```