

DBMS ASSIGNMENT 3

5TH SEM SECTION I

PES1UG19CS579
VISHWAS R

PES1UG19CS548
UTHPAL P

PES1UG19CS534
T R SUDHARSHAN

PROJECT TITLE



ITHUB

IT

Click them this to skip code:

- [Testing simple queries](#)
- [Testing Complex queries](#)
- [Testing Transactions operation](#)
- [Privileges](#)

Complete Working model of the Database application

Demonstrate

 Simple queries

1. User who makes most versions in a project

```
--- ### Task 1: user who makes the most versions in a project###
```

```
create or replace function sq1_main(r rid int)
returns int
language plpgsql
as
$$
declare
    ans integer;
    maxi integer;
begin

    create table temp0 as
    select v_user_id,count(v_user_id) as cnt
    from version
```

```

where v_rem_repo_id = r_rid
group by v_user_id;

select max(cnt)
into maxi
from temp0;

select
    v_user_id
into ans
from temp0
where
    cnt = maxi;

drop table temp0;
return ans;
end;
$$;

```

2. Version belonging to most branches in a project

--- ### Task 2: Version belonging to most branches in a project###

```

create or replace function sq2_main(r_rid int)
returns int
language plpgsql
as
$$
declare
    ans integer;
    maxi integer;
begin

    create table temp0 as
    select vid,count(vid) as cnt
    from holds
    where v_rem_repo_id = r_rid
    group by vid;

    select max(cnt)
    into maxi
    from temp0;

    select vid
    into ans

```

```

from temp0
where cnt = maxi;

drop table temp0;
return ans;
end;
$$;

```

3. File belonging to most versions in a project

```

--- ### Task 3: File belonging to most versions in a project ###
create or replace function sq3_main(r_rid int)
returns int
language plpgsql
as
$$
declare
    ans integer;
    maxi integer;
begin

    create table temp0 as
    select contains_fid,count(contains_fid) as cnt
    from contains
    where contains_v_rem_repo_id = r_rid
    group by contains_fid;

    select max(cnt)
    into maxi
    from temp0;

    select contains_fid
    into ans
    from temp0
    where cnt = maxi;

    drop table temp0;
    return ans;
end;
$$;

```

4. Average size of a file of a specific type

```
--- ### Task 4: Average size of a file with a specified type###
```

```
create or replace function sq4_main(ty varchar)
returns float
language plpgsql
as
$$
declare
    ans float;
begin
    select avg(file_size)
    into ans
    from file
    where type = ty;

    return ans;
end;
$;
```

5. Average salary of a company

```
--- ### Task 5: Average salary of a company ###
```

```
create or replace function sq5_main(c_id int)
returns float
language plpgsql
as
$$
declare
    ans float;
begin
    select avg(salary)
    into ans
    from employs
    where e_company_id = c_id;

    return ans;
end;
$;
```

Testing simple queries

```
C:\Program Files\PostgreSQL\13\bin>psql -U postgres -d github -f "C:\Users\trsud\OneDrive\Desktop\DBMS\project\assignment 3\simple_q.sql"
Password for user postgres:
DROP FUNCTION
DROP FUNCTION
DROP FUNCTION
DROP FUNCTION
DROP FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION

C:\Program Files\PostgreSQL\13\bin>psql -U postgres -d github -f "C:\Users\trsud\OneDrive\Desktop\DBMS\project\assignment 3\simple_q_test_cases.sql"
Password for user postgres:
proj_1_mostcontri
-----
          1
(1 row)

proj_2_baselineversion
-----
          1
(1 row)

proj_3_impfile
-----
         11
(1 row)

avg_py_size
-----
        517.5
(1 row)

comp_2_avg_salary
-----
1733.3333333333333
(1 row)
```

--- ### Task 1: users who makes the most versions in a project ###

```
select sq1_main(1) as proj_1_mostContri;
```

--- ### Task 2: versions belonging to most branches in a project ###

```
select sq2_main(2) as proj_2_baselineVersion;
```

--- ### Task 3: File belonging to most versions in a project ###

```
select sq3_main(3) as proj_3_impFile;
```

--- ### Task 4: Average size of a fiel with a specified type ###

```
select sq4_main('py') as avg_py_size;
```

--- ### Task 5: Average salary of a company ###

```
select sq5_main(2) as comp_2_avg_salary;
```

Complex queries

1) Total amount of money a company spent for a project

$$Project_cost = \sum_{\forall e \in e_p} \frac{e_salary}{e_hours} * e_hours_on_this_project$$

```
---### Task 1: Find the total cost of a project with remote repo  
id (creation of a function) ###  
--- main idea: n_hrs_emp() finds the total hours the employee  
works across all projects  
--- q1_main() uses n_hrs_emp() and calculates the cost of  
employing an employee for this project as  
--- hr_in_this_proj*total_salary/total_hrs_across_all_proj  
--- Then a summation of this value is done for all employees  
across the project
```

```
create or replace function n_hrs_emp(e_id int)  
returns int  
language plpgsql  
as  
$$  
declare  
    ans integer;  
begin  
    with temp0 as(  
        select hours  
        from repository  
        where  
            owns = e_id)  
        select sum(hours)  
        into ans  
        from temp0;  
  
    return ans;  
end;  
$$;
```

```
--- q1_main() computes the overall cost of the project
```

```
--- Accepts project_id(as remote_repo_id),  
--- returns total cost of the project
```

```
create or replace function q1_main(r_rid int)  
  
returns float  
language plpgsql  
as  
$$  
declare  
    ans float;  
begin  
    with temp0 as(  
  
        select  
            owns, hours, salary  
        from  
            repository, employs  
        where  
            remote_rid = r_rid and e_user_id = owns)  
  
        select  
            sum(salary*hours/n_hrs_emp(owns))  
        into  
            ans  
        from  
            temp0;  
  
    return ans;  
end;  
$$;
```

2) Total space for a project

```
--- ### Task 2: To find the total space for the project ###  
--- Relations used: Contains, file
```

```
create or replace function q2_main(r_rid int)  
  
returns int  
language plpgsql  
as  
$$  
declare  
    ans int;  
begin  
    with temp0 as(  
        select distinct contains_fid  
        from contains  
        where contains_v_rem_repo_id = r_rid),  
  
        temp1 as(  
            select file_size, fid  
            from temp0, file  
            where fid = contains_fid)  
  
        select  
            sum(file_size)  
        into ans  
        from  
            temp1;  
  
    return ans;  
end;  
$$;
```


3) Estimated cost for completing project

```
--- ### Task 3: Estimated cost for completing the project ###
create or replace function q3_main(r_rid int)
returns float
language plpgsql
as
$$
declare
    till_cost integer;
    till_perci integer;
    ans integer;
begin

    select q1_main(r_rid)
    into till_cost;

    select progress_bar
    into till_perci
    from project
    where p_rid = r_rid;

    ans = (100-till_perci)/till_perci*till_cost;
    return ans;
end;
$$;
```

4) Project which requires least number of man hours to be completed in a company

```
--- ### Task 4:Project which requires least number of man hours to  
complete ###
```

```
create or replace function n_hrs_project(r rid int)
returns int
language plpgsql
as
$$
declare
    ans float;
begin
    select sum(hours)
    into ans
    from repository
    where remote rid = r rid;

    return ans;
end;
$$;

select 'n_hrs_project function created';

create or replace function n_hrs_left_project(r rid int)
returns float
language plpgsql
as
$$
declare
    n_hrs_p float;
    proj_perci float;
    ans float;
begin
    select progress_bar
    into proj_perci
    from project
    where p rid = r rid;

    select n_hrs_project(r rid)
    into n_hrs_p;

    ans = (100-proj_perci)/proj_perci*n_hrs_p;
```

```
        return ans;
    end;
    $$;

select 'n_hrs_left_project function created';

create or replace function q4_main(c_id int)
returns int
language plpgsql
as
$$

declare
    ans integer;
    min_hrs float;
begin

    select min(n_hrs_left_project(p_rid))
    into min_hrs
    from project
    where p_company_id = c_id;

    select
        p_rid
    into
        ans
    from
        project
    where
        n_hrs_left_project(p_rid) = min_hrs and
        p_company_id = c_id;

    return ans;
end;
$$;
```

5)Largest local repo across all projects

```
--- ### Task 5: Largest local repo across all projects ###
```

```
--- Main idea: i ) identify branches in the repo
```

```
--- ii ) identify all versions in those branches
```

```
--- iii) identify all files in those versions and find
```

```
their total size
```

```
--- iv) using the above 3 steps, find the size of all
```

```
repos and find the largest of them
```

```
create or replace function repo_size(repo_id int)
```

```
returns int
```

```
language plpgsql
```

```
as
```

```
$$
```

```
declare
```

```
    ans integer;
```

```
    rem_repo integer;
```

```
begin
```

```
    with temp0 as(
```

```
    select
```

```
        bid,b_rid
```

```
    from
```

```
        branch
```

```
    where
```

```
        b_rid = repo_id),
```

```
    --- temp1
```

```
    temp1 as(
```

```
    select distinct vid,v_rem_repo_id
```

```
    from holds
```

```
    natural join
```

```
    temp0),
```

```
    --- temp2
```

```
    temp2 as(
```

```
    select distinct contains_fid as fid
```

```
    from contains,temp1
```

```
    where vid = contains_vid and contains_v_rem_repo_id = v_rem_repo_id)
```

```
    select sum(file_size)
```

```
    into ans
```

```
    from file
```

```
    natural join
```

```
    temp2;
```

```
        return ans;
    end;
    $$;

--- select repo_size(2);

select 'repo_size function created';

create or replace function q5_main()
returns int
language plpgsql
as
$$
declare
    ans integer;
    maxi integer;
begin
    select max(repo_size(rid))
    into maxi
    from repository;

    select rid
    into ans
    from repository
    where repo_size(rid) = maxi;

    return ans;
end;
$$;
```

Testing Complex queries

Password for user postgres:

project_1_cost

4100
(1 row)

project_2_cost

2750
(1 row)

project_3_space

2350
(1 row)

project_11_space

2910
(1 row)

project_1_estcost

4100
(1 row)

project_2_estcost

2750
(1 row)

company_1_lstproj

1
(1 row)

company_2_lstproj

3
(1 row)

largest_local_repo

12
(1 row)

--- ### Task 1 : Find the total cost of a project with remote repo id ###

```
select q1_main(1) as project_1_cost;  
select q1_main(2) as project_2_cost;
```

--- ### Task 2: To find the total space for the project ###

```
select q2_main(3) as project_3_space;  
select q2_main(11) as project_11_space;
```

--- ### Task 3: Estimated cost for completing the project ###

```
select q3_main(1) as project_1_estCost;  
select q3_main(2) as project_2_estCost;
```

--- ### Task 4: Project which requires least number of man hours to complete in a company

####

```
select q4_main(1) as company_1_lstProj;  
select q4_main(2) as company_2_lstProj;
```

--- ### Task 5: Largest local repo across all projects ###

```
select q5_main() as largest_local_repo;
```

Transactional Operations:

Update salary:

Inputs: User_ID, Company_ID, New Salary

Output: Updated salary value.

In Employs Table, update salary.

```
-----# fucntion to update salary #
create or replace function update_salary(empID int,cmpID int, newSalary int)
returns int as $$
declare
    updated_val int;
begin
    UPDATE Employs
    SET Salary=newSalary
    WHERE E_user_ID=empID and E_Company_ID=cmpID;
    Select Salary into updated_val from Employs where E_user_ID=empID and
E_Company_ID=cmpID;
    return updated_val;
end;
$$ language plpgsql;
```

Add new Employee to company:

Inputs: Developer_User_ID, Company_ID, Salary

Output: - In employs table, add new entry.

```
create or replace function recruiting (empid int , CmpID int , salary numeric(10,2))
returns int as $$
declare
    r int;
begin
    IF ((SELECT(CmpID) IN (SELECT Company_ID FROM Company)) and
(SELECT(empid) IN (SELECT User_ID FROM Developer)) and ((SELECT(empid) NOT
IN (SELECT E_User_ID FROM Employs)) and (SELECT(CmpID) NOT IN (SELECT
E_Company_ID FROM Employs)))) ) THEN
        INSERT INTO Employs values (empid,CmpID,salary);
        RETURN 1;
    ELSE
        RETURN -1;
    END IF;
end;
$$ language plpgsql;
```

Create Developer:

```
drop function if exists create_developer (Pass character varying);

create or replace function create_developer(Pass character varying)
returns int as $$
declare
    dev int;
begin
    SELECT (MAX(User_ID)+1) into dev from Developer;
    INSERT INTO Developer(User_ID, Password)
    VALUES (dev, pass);
    return dev;
end;
$$ language plpgsql;
```

Create Company:

```
create or replace function create_company (comp_des varchar,comp_loc varchar)
returns int as $$
declare
    comp int;
begin
    SELECT (MAX(Company_ID)+1) into comp from Company;
    INSERT INTO Company (Company_ID, Company_Description ,Location)
    VALUES (comp, comp_des,comp_loc);
    return comp;
end;

$$ language plpgsql;
```

Add new Project:

[Add new Remote Repository as a result of it]

Inputs: Company_ID ,Description

Returns : Repo ID of the new remote repo created.

Process:

Verify from company table if the given CompanyID exists.

Add new repository having P_RID = max(Repo(RID))+1

Create new master branch in the newly created repository.

Add new Project entry- {P_RID, description, 0, Company_ID}

Return P_RID;

```
---# function to create new project #
```

```
create or replace function create_project(CmpID int, Description varchar(200))
returns int as $$
declare
```



```

    New_P_RID int;
begin
    IF (SELECT(CmpID) IN (SELECT Company_ID FROM Company)) THEN
        SELECT (MAX(RID)+1) into New_P_RID from Repository;
        INSERT INTO Repository values (New_P_RID);
        INSERT INTO Project values (New_P_RID, Description, 0,CmpID);
        INSERT INTO Branch values (1,New_P_RID,'23:05:06',null,null);
        RETURN New_P_RID;
    ELSE
        RETURN -1;
    END IF;
end;
$$ language plpgsql;

```

Create Branch:

Inputs: RepoID, Parent BranchID, UserID

Output: New Branch ID of created branch

Process:

Verify if the repo belongs to the UserID.

Find remote repo ID from local repoID.

NewBranchID = MAX(BranchID in remoteRepoID)+1

Add Branch entry for Remote repo.

Add branch entry for local repo.

Return NewBranchID

```

---# function to create branch #
create or replace function create_branch(devID int, RepoID int, Parent_BranchID int)
returns int as $$
declare
    New_Branch_ID int;
    Remote_Repo_ID int;
begin
    IF (SELECT((SELECT COUNT(*) FROM Repository where RID=RepoID and
Owns=devID)>0)) THEN
        SELECT Remote_RID into Remote_Repo_ID FROM Repository where
RID=RepoID and Owns=devID;
        SELECT (MAX(BID)+1) into New_Branch_ID from Branch where
B_RID=Remote_Repo_ID;
        INSERT INTO Branch values
(New_Branch_ID,Remote_Repo_ID,'23:05:06',Parent_BranchID,Remote_Repo_ID);
        INSERT INTO Branch values
(New_Branch_ID,RepoID,'23:05:06',Parent_BranchID,RepoID);
        RETURN New_Branch_ID;
    ELSE
        RETURN -1;
    END IF;
end;
$$ language plpgsql;

```

Commit:

Inputs:

CreateNewVersion(if 1 create New version and commit elseif 0 commit to latest version),

BranchID, RepoID,

FileName, file type, Filecontents, file size.

Output:

VID if commit successful else -1.

Process:

If CreatenewVersion = 1 ,

then Get latest version of the remote repo id.

Create new version newVID using latestVersion+1.

Else,

Get latest version ID and set the variable newVID=latestVID.

After getting new version:

Generate new FileID using MAX(FID)+1

Append all file related details into file table for newFID and update the Contains table with the new fileID generated

---# function to commit #

```
create or replace function Commit(Create_NewVersion int, Commit_into_BID int,
Commit_into_RepoID int, commit_file_name varchar(30), commit_file_type varchar(10),
commit_file_contains text, commit_file_size int)
returns int as $$
declare
    latest_existing_version integer;
    New_Version integer;
    REM_RepoID integer;
    MAX_RANK1 integer;
    New_FID integer;
begin
    SELECT Remote_RID into REM_RepoID FROM Repository where
RID=Commit_into_RepoID;
    IF (Create_NewVersion=1) THEN -- create a new version
        SELECT MAX(VID)+1 INTO New_Version from Version where
V_rem_repo_ID=REM_RepoID;
        SELECT MAX(Rank) into MAX_RANK1 FROM Holds where
BID=Commit_into_BID and B_RID=Commit_into_RepoID;
        SELECT VID into latest_existing_version FROM Holds where
BID=Commit_into_BID and B_RID=Commit_into_RepoID and rank=MAX_RANK1;
        INSERT INTO Version values (New_Version,REM_RepoID,(SELECT
V_User_ID from version where VID=latest_existing_version and
V_rem_repo_ID=REM_RepoID));
        INSERT INTO Holds values
(Commit_into_BID,Commit_into_RepoID,New_Version,REM_RepoID,(MAX_RANK1+1));

    ELSIF (Create_NewVersion=0) THEN -- add file to latest existing version.
        SELECT MAX(VID) INTO New_Version from Version where
V_rem_repo_ID=REM_RepoID;
```

```

ELSE -- if Create_NewVersion not either 1 or 0, return -1 indicating wrong input
    RETURN -1;
END IF;

SELECT MAX(FID)+1 INTO New_FID FROM File;
INSERT INTO File values
(New_FID, commit_file_name, commit_file_type, commit_file_contains,
commit_file_size);
INSERT INTO Contains values (New_Version, REM_RepoID, New_FID);
RETURN New_Version; --return version number of the commit.
end;
$$ language plpgsql;

```

Merge:

Inputs: Merge1_BID, Merge2_BID, RepoID

Output: new version in Merge1 branch.

Process:

1. Verify the two branches are in the given repo;
 - check if Merge1_BID, Merge2_BID in branch where B_RID=RepoID
2. Find latest versions of both branches.
 - Get latest versions (VID+V_Rem_repo_ID) from MAX(Rank) in holds table where BID= Merge1_BID and Merge2_BID
3. Get files in both the version:
 - Select Contains table where VID= latest versions of the two branches.
4. Groupby filename and create a view GROUPED_FILES.
5. Insert New version into Branch1 and get New_VID.
6. for each group in GROUPED_FILES,
 - insert the file with highest size into the new version. [update contains table for each new file added into version.] (no changes in File table!)

```

---# function to merge #
create or replace function Merge(Merge2_BID int, Merge1_BID int, RepoID int) ---#
Merge Branch2 into branch1
returns int as $$
declare
    REM_RepoID integer;
    New_VID integer;
    MAX_RANK1 integer; -- max rank of latest VID of branch 1
    MAX_RANK2 integer; -- max rank of latest VID of branch 2
    B1_latest_VID integer;
    B2_latest_VID integer;
    Max_Size integer;
    rec RECORD;
begin
    IF (SELECT(Merge1_BID) NOT IN (SELECT BID FROM Branch WHERE
B_RID=RepoID)) THEN
        RETURN -1;

```

```

        END IF;
        IF (SELECT(Merge2_BID) NOT IN (SELECT BID FROM Branch WHERE
B_RID=RepoID)) THEN
            RETURN -1;
        END IF;
        SELECT Remote_RID into REM_RepoID FROM Repository where RID=RepoID;
        SELECT MAX(Rank) into MAX_RANK1 FROM Holds where BID=Merge1_BID and
B_RID=RepoID;
        SELECT MAX(Rank) into MAX_RANK2 FROM Holds where BID=Merge2_BID and
B_RID=RepoID;
        SELECT VID into B1_latest_VID FROM Holds where BID=Merge1_BID and
B_RID=RepoID and rank=MAX_RANK1;
        SELECT VID into B2_latest_VID FROM Holds where BID=Merge2_BID and
B_RID=RepoID and rank=MAX_RANK2;

        SELECT MAX(VID)+1 INTO New_VID from Version where
V_rem_repo_ID=REM_RepoID;
        INSERT INTO Version values (New_VID,REM_RepoID,(SELECT V_User_ID from
version where VID=B1_latest_VID and V_rem_repo_ID=REM_RepoID));
        INSERT INTO Holds values (Merge1_BID,RepoID,New_VID,REM_RepoID,
(MAX_RANK1+1));

        CREATE table ALL_Files_to_be_merged AS (SELECT
f.fid,f.file_name,f.file_contains,f.file_size,f.type from File f,Contains c where
c.Contains_V_rem_repo_ID=REM_RepoID and (c.Contains_VID=B1_latest_VID or
c.Contains_VID=B2_latest_VID) and f.FID=c.Contains_FID);
        FOR rec in (SELECT DISTINCT file_name FROM ALL_Files_to_be_merged)
LOOP
            SELECT MAX(file_size) into Max_Size from ALL_Files_to_be_merged where
file_name=rec.file_name;
            INSERT INTO Contains values
            (New_VID,REM_RepoID,(Select DISTINCT FID from
ALL_Files_to_be_merged where file_name=rec.file_name and file_size=Max_Size));
        END LOOP;
        DROP table ALL_Files_to_be_merged;
        return New_VID; --return version number the new commit creates.
    end;
    $$ language plpgsql;

```

Pull:

Input: local repo id , user id;

Output : 1 on successful pull / 0 on error

Process :

Handle commits - find out new commits in remote repo and add then to local repo.

Handle branches - find out new branches in remote repo and add then to local repo.

```
create or replace function pull (devID int, RepoID int)
returns int as $$
declare
    Remote_Repo_ID int;
    c int;
    c1 int;
    t time;
begin
    t='09:15:08';
    IF (SELECT (SELECT COUNT(*) FROM Repository where RID=RepoID and
Owns=devID) >0 ) THEN
        SELECT Remote_RID into Remote_Repo_ID FROM Repository where
RID=RepoID and Owns=devID;

        INSERT INTO Branch
        select distinct h1.BID ,RepoID as B_RID, t as
timestamp ,h1.parent_BID,h1.parent_B_RID from (SELECT * FROM branch WHERE
b_rid=Remote_Repo_ID) h1, (Select b.bid from (SELECT * FROM Holds WHERE
b_rid=Remote_Repo_ID) b,(SELECT vid FROM Holds WHERE b_rid=Remote_Repo_ID
EXCEPT SELECT vid FROM Holds WHERE b_rid=RepoID) v where v.vid=b.vid EXCEPT
select BID from branch where B_rid=RepoID) h2 where h1.bid=h2.bid;

        INSERT INTO Holds (bid , b_rid , vid , v_rem_repo_id ,rank)
        Select b.bid,RepoID as B_RID,b.vid,b.v_rem_repo_id,b.rank from (SELECT *
FROM Holds WHERE b_rid=Remote_Repo_ID) b,(SELECT vid FROM Holds WHERE
b_rid=Remote_Repo_ID EXCEPT SELECT vid FROM Holds WHERE b_rid=RepoID) v
where v.vid=b.vid;
        return 1;
    ELSE
        RETURN -1;
    END IF;
end;
$$ language plpgsql;
```

Push:

Input: local repo id , user id;

Output : 1 on successful pull / 0 on error

Process :

Handle commits - find out new commits in local repo and add then to remote repo.

Handle branches - no need to handle branches as in remote repo will contain all branches .

```
create or replace function push (devID int, RepoID int)
returns int as $$
declare
    Remote_Repo_ID int;
    c int;
    c1 int;
    t time;
begin
    t='09:15:08';
    IF (SELECT (SELECT COUNT(*) FROM Repository where RID=RepoID and
Owns=devID) >0 ) THEN
        SELECT Remote_RID into Remote_Repo_ID FROM Repository where
RID=RepoID and Owns=devID;
        INSERT INTO Holds (bid , b_rid , vid , v_rem_repo_id ,rank)
        Select b.bid,Remote_Repo_ID as B_RID,b.vid,b.v_rem_repo_id,b.rank from
(SELECT * FROM Holds WHERE b_rid=RepoID) b,(SELECT vid FROM Holds WHERE
b_rid=RepoID EXCEPT SELECT vid FROM Holds WHERE b_rid=Remote_Repo_ID) v
where v.vid=b.vid;
        return 1;
    ELSE
        RETURN -1;
    END IF;
end;
$$ language plpgsql;
```

Testing Transactions operation

Create Developer:

```
github=# select * from developer;
 user_id | password
-----+-----
      1 | password1
      2 | password2
      3 | password3
      4 | password4
      5 | password5
      6 | password6
      7 | password7
      8 | password8
(8 rows)

github=# select create_developer ('password_new');
 create_developer
-----
              9
(1 row)

github=# select * from developer;
 user_id | password
-----+-----
      1 | password1
      2 | password2
      3 | password3
      4 | password4
      5 | password5
      6 | password6
      7 | password7
      8 | password8
      9 | password_new
(9 rows)
```

Create Company:

```
github=# select * from company;
 company_id | company_description | location
-----+-----+-----
          1 | Comp_desc1         | loc1
          2 | Comp_desc2         | loc2
(2 rows)

github=# select create_company ('Comp_new_desc','loc_new');
 create_company
-----
              3
(1 row)

github=# select * from company;
 company_id | company_description | location
-----+-----+-----
          1 | Comp_desc1         | loc1
          2 | Comp_desc2         | loc2
          3 | Comp_new_desc      | loc_new
(3 rows)
```

Add new Employee to company:

```
github=# select * from Employs;
 e_user_id | e_company_id | salary
-----+-----+-----
          1 |             1 | 1500.00
          2 |             1 | 1400.00
          3 |             1 | 1200.00
          5 |             2 | 1700.00
          6 |             2 | 1600.00
          7 |             2 | 1900.00
(6 rows)

github=# select recruiting(9,3,35000);
 recruiting
-----
          1
(1 row)

github=# select * from Employs;
 e_user_id | e_company_id | salary
-----+-----+-----
          1 |             1 | 1500.00
          2 |             1 | 1400.00
          3 |             1 | 1200.00
          5 |             2 | 1700.00
          6 |             2 | 1600.00
          7 |             2 | 1900.00
          9 |             3 | 35000.00
(7 rows)
```

Add new Project:

```
Command Prompt - psql -U postgres
You are now connected to database "github" as user "postgres".
github=# select * from project;
 p_rid | description | progress_bar | p_company_id
-----+-----+-----+-----
       1 | proj_desc_1 |          50 |             1
       2 | proj_desc_2 |          40 |             2
       3 | proj_desc_3 |          60 |             2
      11 | proj_desc_4 |          70 |             2
(4 rows)

github=# Select * from Repository;
 rid | remote_rid | owns | hours
-----+-----+-----+-----
     1 |           |     | 
     2 |           |     | 
     3 |           |     | 
    11 |           |     | 
     4 |           1 |     1 |    10
     5 |           1 |     2 |    12
     6 |           1 |     3 |     4
     7 |           2 |     5 |    11
     8 |           2 |     7 |     6
     9 |           3 |     6 |    12
    10 |           3 |     5 |    11
    12 |          11 |     8 |    14
(12 rows)

github=# SELECT Create_project(1,'Company 1- Second Project');
 create_project
-----
          13
(1 row)
```

```
Command Prompt - psql -U postgres

github=# SELECT Create_project(1,'Company 1- Second Project');
 create_project
-----
          13
(1 row)

github=# select * from project;
 p_rid | description | progress_bar | p_company_id
-----+-----+-----+-----
       1 | proj_desc_1 |          50 |             1
       2 | proj_desc_2 |          40 |             2
       3 | proj_desc_3 |          60 |             2
      11 | proj_desc_4 |          70 |             2
      13 | Company 1- Second Project |          0 |             1
(5 rows)

github=# Select * from Repository;
 rid | remote_rid | owns | hours
-----+-----+-----+-----
     1 |           |     | 
     2 |           |     | 
     3 |           |     | 
    11 |           |     | 
     4 |           1 |     1 |    10
     5 |           1 |     2 |    12
     6 |           1 |     3 |     4
     7 |           2 |     5 |    11
     8 |           2 |     7 |     6
     9 |           3 |     6 |    12
    10 |           3 |     5 |    11
    12 |          11 |     8 |    14
    13 |           |     | 
(13 rows)
```


Update salary:

```
Command Prompt - psql -U postgres
github=# select * from employs;
 e_user_id | e_company_id | salary
-----+-----+-----
          1 |              1 | 1500.00
          2 |              1 | 1400.00
          3 |              1 | 1200.00
          5 |              2 | 1700.00
          6 |              2 | 1600.00
          7 |              2 | 1900.00
(6 rows)

github=# SELECT update_salary(7, 2, 2000);
update_salary
-----
         2000
(1 row)

github=# select * from employs;
 e_user_id | e_company_id | salary
-----+-----+-----
          1 |              1 | 1500.00
          2 |              1 | 1400.00
          3 |              1 | 1200.00
          5 |              2 | 1700.00
          6 |              2 | 1600.00
          7 |              2 | 2000.00
(6 rows)

github=#
```

Create Branch:

```
Command Prompt - psql -U postgres
You are now connected to database "github" as user "postgres".
github=# Select * from Branch where b_rid=1 or B_rid=4;
 bid | b_rid | timestamp | parent_bid | parent_b_rid
-----+-----+-----+-----+-----
    1 |      1 | 04:05:06 |           | 
    1 |      4 | 04:05:06 |           | 
    2 |      1 | 04:05:07 |           | 
    3 |      1 | 04:05:08 |           | 
    4 |      1 | 04:05:09 |           | 
    2 |      4 | 04:05:07 |           | 
(6 rows)

github=# --create_branch(devID int, RepoID int, Parent_BranchID int)
github=# ;
github=# SELECT create_branch(1,4,1);
create_branch
-----
          5
(1 row)

github=# Select * from Branch where b_rid=1 or B_rid=4;
 bid | b_rid | timestamp | parent_bid | parent_b_rid
-----+-----+-----+-----+-----
    1 |      1 | 04:05:06 |           | 
    1 |      4 | 04:05:06 |           | 
    2 |      1 | 04:05:07 |           | 
    3 |      1 | 04:05:08 |           | 
    4 |      1 | 04:05:09 |           | 
    2 |      4 | 04:05:07 |           | 
    5 |      1 | 23:05:06 |           | 
    5 |      4 | 23:05:06 |           | 
(8 rows)

github=# -- new branch 5 gets created in repo 4 and its remote repo 1 also.
github=#
```

Merge :

Command Prompt - psql -U postgres

```
github=#
github=# -- merge branch 2 into branch1 (master branch) in repository 4:
github=#
github=# -- files present in latest version of master branch
github=# SELECT * from File f,Contains c where c.Contains_V_rem_repo_ID=1 and (c.Contains_VID=3) and f.FID=c.Contains_FID;
fid | file_name | type | file_contains | file_size | contains_vid | contains_v_rem_repo_id | contains_fid
-----+-----+-----+-----+-----+-----+-----+-----
 1 | read      | md   | file_desc1    |    100    |          3 |          1             |          1
 3 | sub       | c    | file_desc3    |    540    |          3 |          1             |          3
 4 | subh      | h    | file_desc4    |    300    |          3 |          1             |          4
(3 rows)
```

```
github=# -- files present in latest version of branch 2
github=# SELECT * from File f,Contains c where c.Contains_V_rem_repo_ID=1 and (c.Contains_VID=2) and f.FID=c.Contains_FID;
fid | file_name | type | file_contains | file_size | contains_vid | contains_v_rem_repo_id | contains_fid
-----+-----+-----+-----+-----+-----+-----+-----
 1 | read      | md   | file_desc1    |    100    |          2 |          1             |          1
 2 | sub       | c    | file_desc2    |    500    |          2 |          1             |          2
 5 | subh      | h    | file_desc5    |    310    |          2 |          1             |          5
(3 rows)
```

```
github=#
github=# --Merge(Merge2_BID int, Merge1_BID int, RepoID int)
github=# Select Merge(2,1,4);
merge
-----
    6
(1 row)
```

```
github=# SELECT * from File f,Contains c where c.Contains_V_rem_repo_ID=1 and (c.Contains_VID=6) and f.FID=c.Contains_FID;
fid | file_name | type | file_contains | file_size | contains_vid | contains_v_rem_repo_id | contains_fid
-----+-----+-----+-----+-----+-----+-----+-----
 1 | read      | md   | file_desc1    |    100    |          6 |          1             |          1
 2 | sub       | c    | file_desc2    |    500    |          6 |          1             |          2
 5 | subh      | h    | file_desc5    |    310    |          6 |          1             |          5
(3 rows)
```

```
github=#
```

Commit:

```
Command Prompt - psql -U postgres

github=#
github=# -- commit a file into branch 1 of repository 9:
github=#
github=# --current versions in branch 1 or repo 9:
github=#
github=# SELECT * from holds where BID=1 and B_RID=9;
bid | b_rid | vid | v_rem_repo_id | rank
-----+-----+-----+-----+-----
  1  |    9  |   1  |             3  |    1
(1 row)

github=#
github=# -- Commit(Create_NewVersion , Commit_into_BID , Commit_into_RepoID , commit_file_name , commit_file_type, commit_file_contents, commit_file_size)
github=#
github=# SELECT Commit(1, 1, 9, 'new_file', 'txt', 'Contents 1234!', 100 );
commit
-----
      6
(1 row)

github=#
github=# -- versions present in branch 1 or repo 9 after committing new file :
github=#
github=# SELECT * from holds where BID=1 and B_RID=9;
bid | b_rid | vid | v_rem_repo_id | rank
-----+-----+-----+-----+-----
  1  |    9  |   1  |             3  |    1
  1  |    9  |   6  |             3  |    2
(2 rows)

github=# -- here we see new version (vid 6) getting added.
github=#

github=# SELECT * from File f,Contains c where c.Contains_V_rem_repo_ID=3 and (c.Contains_VID=6) and f.FID=c.Contains_FID;
fid | file_name | type | file_contains | file_size | contains_vid | contains_v_rem_repo_id | contains_fid
-----+-----+-----+-----+-----+-----+-----+-----
  22 | new_file  | txt  | Contents 1234! |      100 |           6 |                 3 |          22
(1 row)

github=# -- files present in the new commit having (vid = 6)
```

Pull:

```
github=# SELECT * FROM Holds WHERE b_rid=5;
```

bid	b_rid	vid	v_rem_repo_id	rank
1	5	1	1	1
2	5	2	1	1
4	5	4	1	1

(3 rows)

```
github=# SELECT * FROM branch WHERE b_rid=5;
```

bid	b_rid	timestamp	parent_bid	parent_b_rid
1	5	04:05:06		
2	5	04:05:07	1	5
4	5	04:05:09	2	5

(3 rows)

```
github=# select pull(2,5);
```

```
pull
-----
1
(1 row)
```

```
github=# SELECT * FROM Holds WHERE b_rid=5;
```

bid	b_rid	vid	v_rem_repo_id	rank
1	5	1	1	1
2	5	2	1	1
4	5	4	1	1
1	5	3	1	2

(4 rows)

```
github=# SELECT * FROM branch WHERE b_rid=5;
```

bid	b_rid	timestamp	parent_bid	parent_b_rid
1	5	04:05:06		
2	5	04:05:07	1	5
4	5	04:05:09	2	5

(3 rows)

Push:

```
github=# SELECT * FROM Holds WHERE b_rid=1;
bid | b_rid | vid | v_rem_repo_id | rank
-----+-----+-----+-----+-----
1 | 1 | 1 | 1 | 1
2 | 1 | 2 | 1 | 1
1 | 1 | 3 | 1 | 2
(3 rows)
```

```
github=# SELECT * FROM Holds WHERE b_rid=5;
bid | b_rid | vid | v_rem_repo_id | rank
-----+-----+-----+-----+-----
1 | 5 | 1 | 1 | 1
2 | 5 | 2 | 1 | 1
4 | 5 | 4 | 1 | 1
1 | 5 | 3 | 1 | 2
(4 rows)
```

```
github=# push(2,5);
ERROR: syntax error at or near "push"
LINE 1: push(2,5);
        ^
```

```
github=# select push(2,5);
push
-----
1
(1 row)
```

```
github=# SELECT * FROM Holds WHERE b_rid=1;
bid | b_rid | vid | v_rem_repo_id | rank
-----+-----+-----+-----+-----
1 | 1 | 1 | 1 | 1
2 | 1 | 2 | 1 | 1
1 | 1 | 3 | 1 | 2
4 | 1 | 4 | 1 | 1
(4 rows)
```

Multiple users with different access privilege levels for different parts of the database should be created.

Types of users:

1. Database Admin (Postgres)
2. Company
3. Developers
4. Viewers.

Privileges provided to different users:

1. Company:

SELECT (view) – all tables except Developer(passwords).
INSERT – Project, Employs, Repository Table
UPDATE (edit) – Company(Description, Location), Project(Description, Progress_bar), Employs(Salary) Table
DELETE – rows in Project, Employs, Repository Table

2. Developers –

SELECT (view) – all tables.
INSERT – Repository, Branch, holds, version, Contains, File Table
UPDATE (edit) – Repository, Branch, holds, version, Contains, File Table
DELETE – rows in Repository, Contains, (Branch, holds, version, Contains -- used only if full repo is deleted), File Table

3. Viewers (Eg. Auditors)

SELECT(View) – all tables except Developer(passwords), file(contains).

```
\c github;
```

```
CREATE USER Company_User;  
CREATE USER Developer_User;  
CREATE USER Viewer;
```

```
---# Company_User permissions#
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO Company_User;  
REVOKE SELECT (Password) ON Developer FROM Company_User;  
GRANT INSERT ON Project,Employs,Repository TO Company_User;  
GRANT UPDATE (Company_Description,Location) ON company TO  
Company_User;  
GRANT UPDATE (Description,progress_bar) ON project TO Company_User;  
GRANT UPDATE (salary) ON employs TO Company_User;  
GRANT DELETE ON Project, Employs,Repository TO Company_User;  
  
---# Developer_User permissions#  
GRANT SELECT ON ALL TABLES IN SCHEMA public TO Developer_User;  
GRANT INSERT,UPDATE ON Repository,Branch,Holds,Version,Contains,File TO  
Developer_User;  
GRANT DELETE ON Repository,Branch,Holds,Version,Contains,File TO  
Developer_User;  
  
GRANT SELECT ON ALL TABLES IN SCHEMA public TO Viewer;  
REVOKE SELECT (Password) ON Developer FROM Viewer;  
REVOKE SELECT (file_contains) ON File FROM Viewer;
```

```
vishwas@Vishwass-MBP I5_Assisgment-02 % psql postgres -f plivileges.sql  
You are now connected to database "github" as user "vishwas".  
CREATE ROLE  
CREATE ROLE  
CREATE ROLE  
GRANT  
REVOKE  
GRANT  
GRANT  
GRANT  
GRANT  
GRANT  
GRANT  
GRANT  
GRANT  
GRANT  
GRANT  
REVOKE  
REVOKE  
vishwas@Vishwass-MBP I5_Assisgment-02 %
```

Indicate contribution of each member clearly and approximate time spent in hours for the respective activity.

T R SUDHARSHAN

Implemented 5 simple query and 5 complex query
Hours spent 25 - 30 hrs

UTHPAL P

Implemented half of the transaction (merge , commit, etc ...)
Implemented access privilege levels
Hours spent 25 - 30 hrs

VISHWAS R

Implemented another half of the transaction
(Push ,Pull ,etc...)
Report work
Hours spent 25 - 30 hrs

We spent a lot of time in modifying the ER diagram as there very many conditions to be handled in our project.