LAB 9

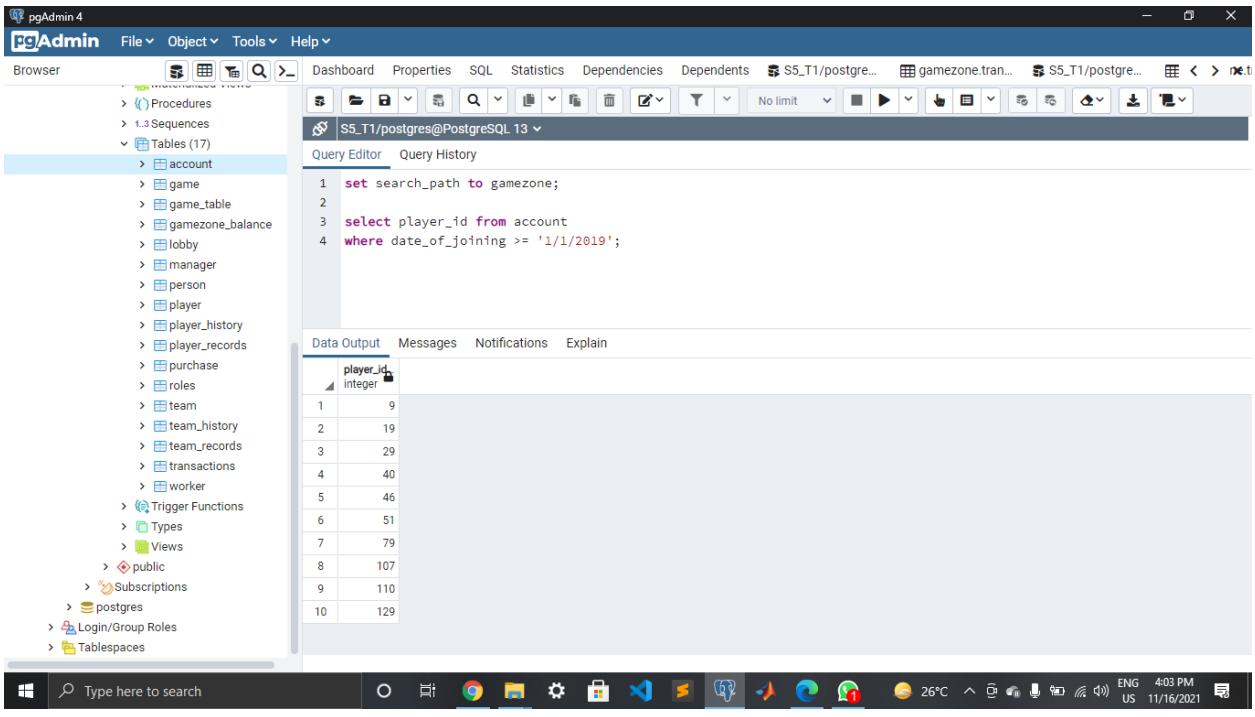
SQL Queries

Thakkar Amit (201901038)
Makwana Jigar (201901428)
Prajapati Parth (201901429)
Gohil Arpit (201901471)

SQL Queries:

1. **List all the players who have joined the gamezone on or after 1/1/2019.**

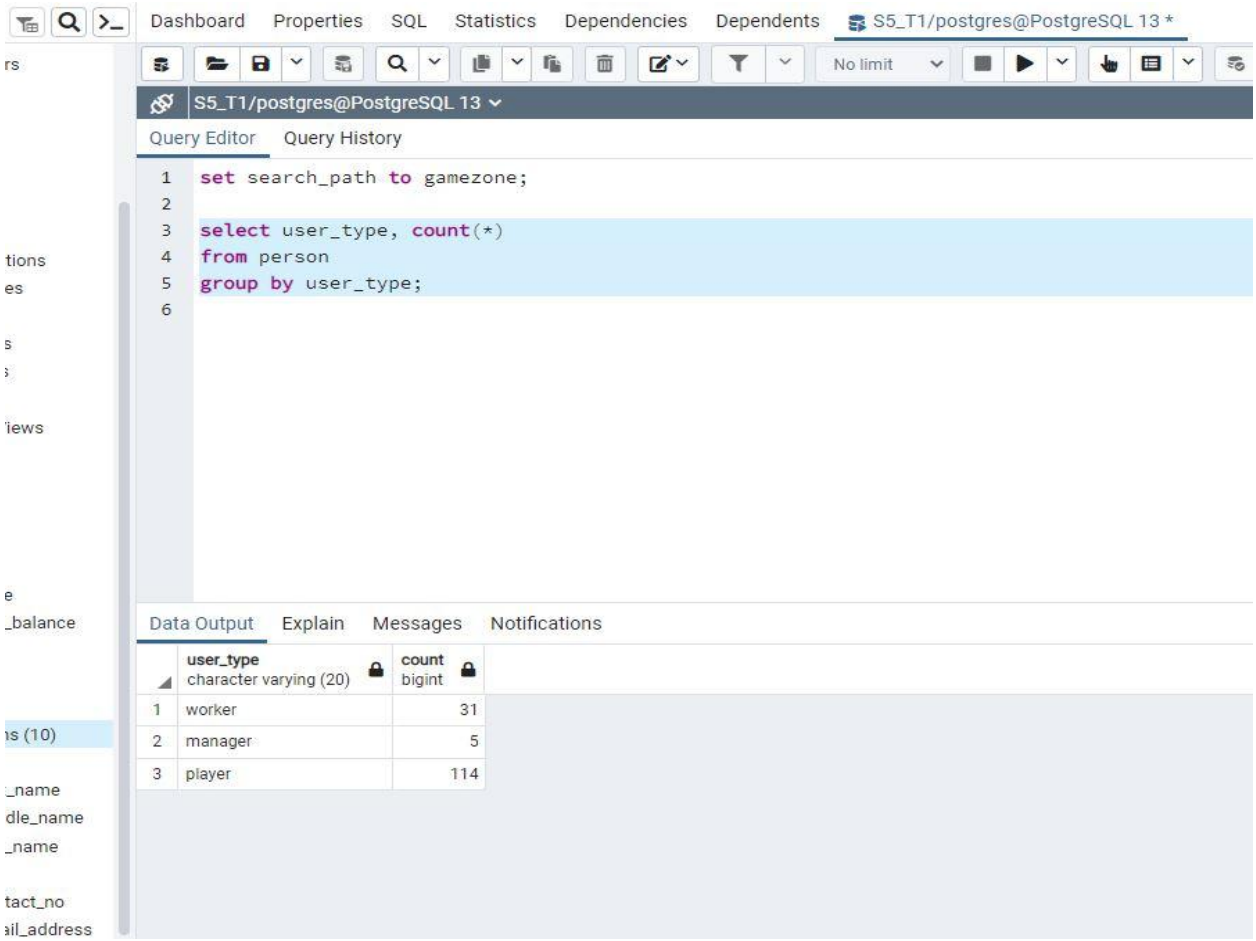
```
select player_id from account
where date_of_joining >= '1/1/2019';
```



No of tupel = 10

2. **Count the number of players, managers and employees present in the person table.**

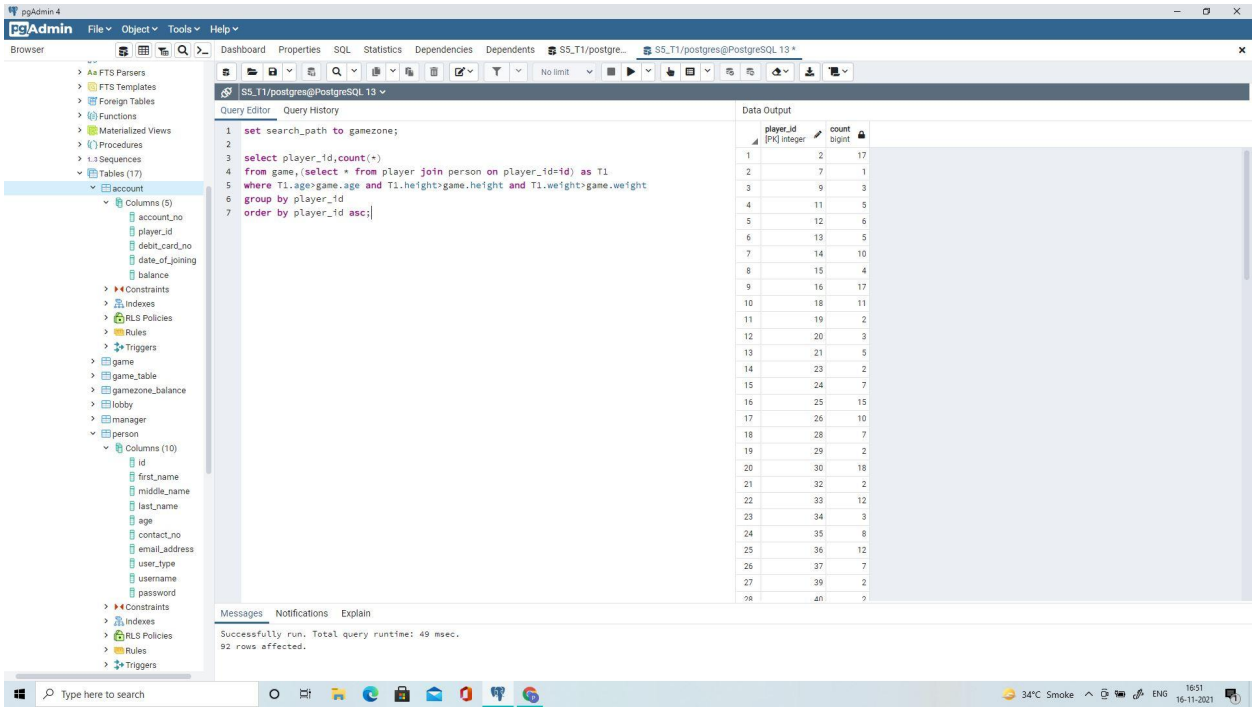
```
select user_type, count(*)
from person
group by user_type;
```



No of tuple = 3

3. **Count the number of games that player can play.**

```
select player_id,count(*)
from game,(select * from player join person on player_id=id) as T1
where T1.age>game.age and T1.height>game.height and
T1.weight>game.weight
group by player_id
order by player_id asc;
```

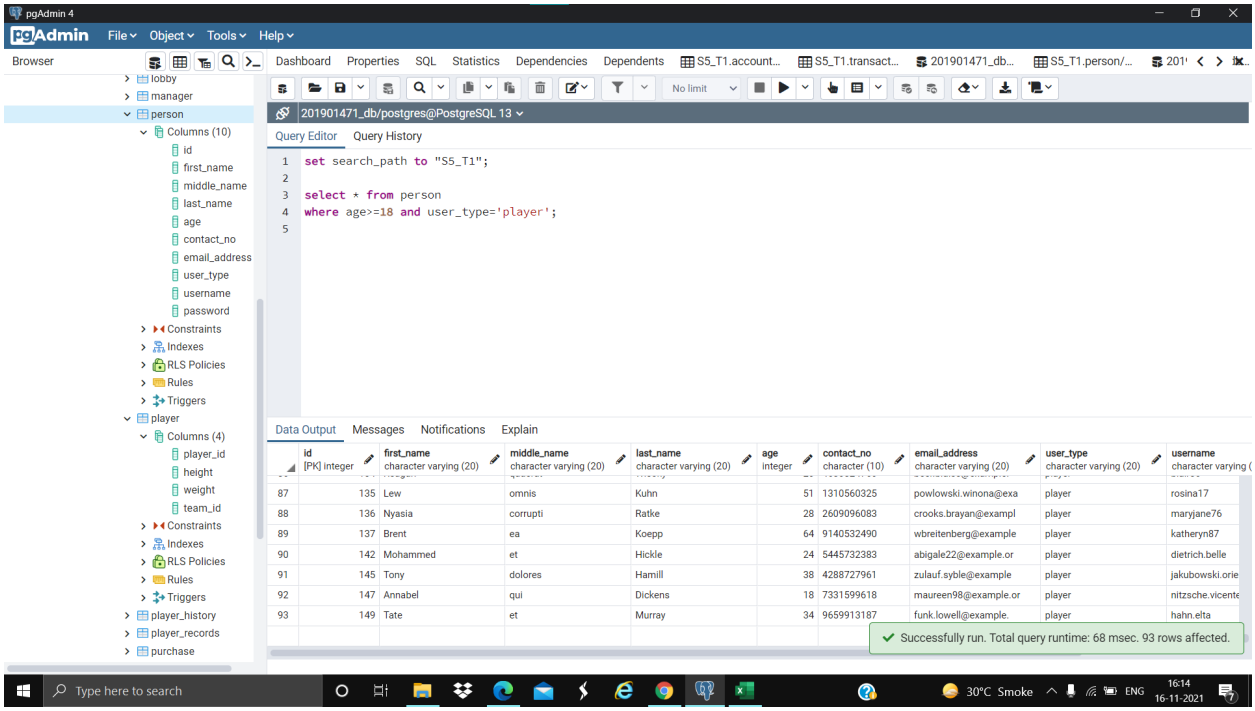


No of tuple = 92

4. List all players whose age is greater than 18.

Select * from person

Where age>=18 and user_type = ‘player’;

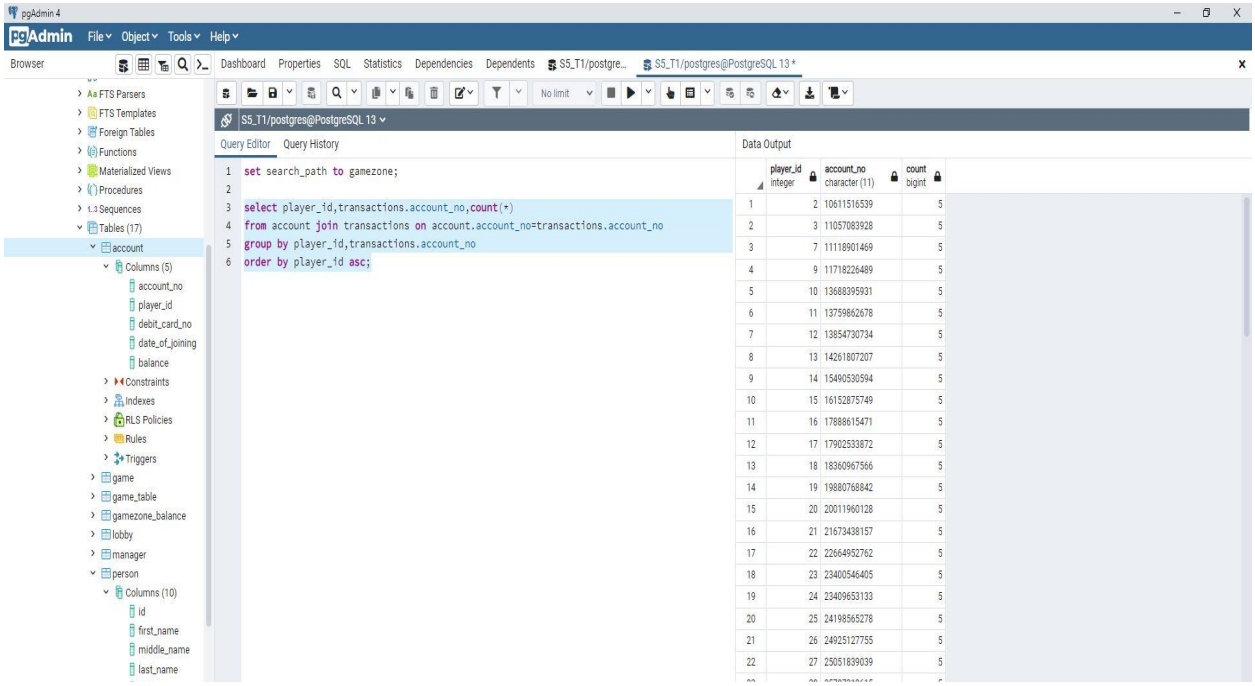


id [PK] integer	first_name character varying (20)	middle_name character varying (20)	last_name character varying (20)	age integer	contact_no character (10)	email_address character varying (20)	user_type character varying (20)	username character varying
87	135	Lew	omnis	Kuhn	51	1310560325	powlowski.winona@exa	player	rosina17
88	136	Nyasia	corrupti	Ratke	28	2609096083	crooks.brayan@exampl	player	maryjane76
89	137	Brent	ea	Koepp	64	9140532490	wbreitenberg@example	player	katheryn87
90	142	Mohammed	et	Hickle	24	5445732383	abigale22@example.or	player	dietrich.belle
91	145	Tony	dolores	Hamill	38	4288727961	zulauf.syble@example	player	jakubowski.orie
92	147	Annabel	qui	Dickens	18	7331599618	maureen98@example.or	player	nitzsche.vicente
93	149	Tate	et	Murray	34	9659913187	funk.lowell@example.	player	hahn.elta

No. of Tuples : 93

5. **Count the number of transactions done by each player.**

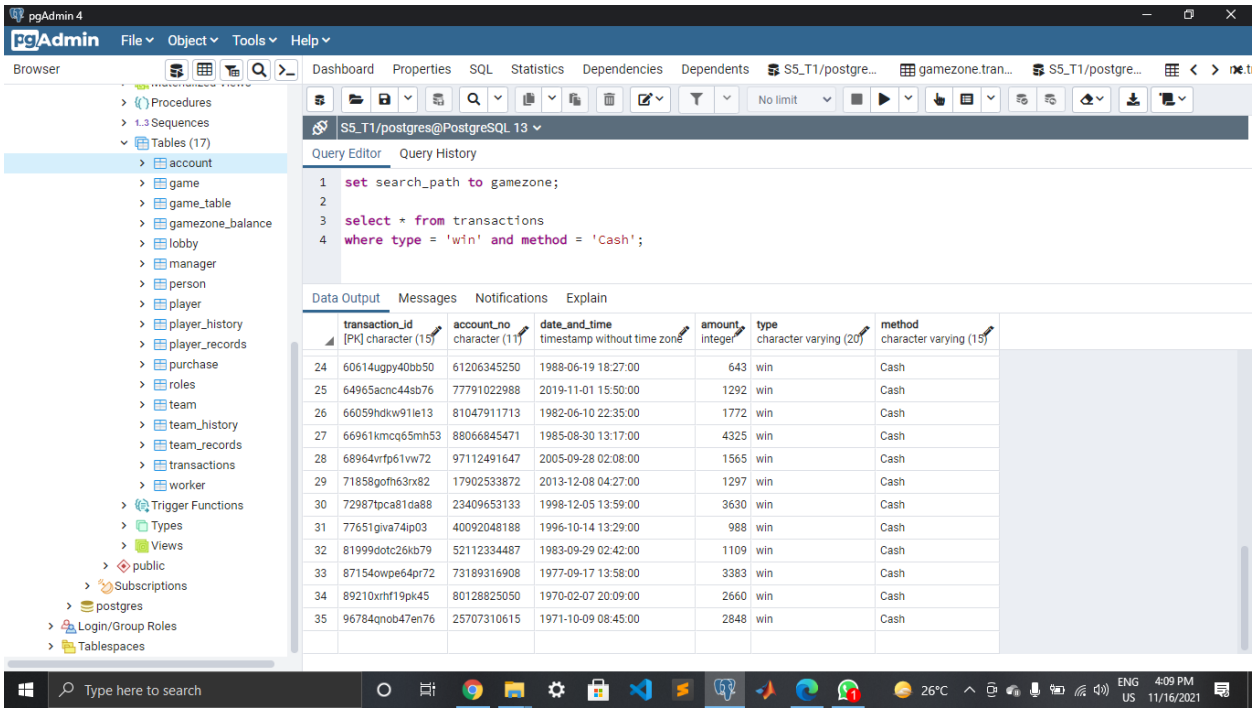
```
select player_id,transactions.account_no, count(*)
from account join transactions on
account.account_no=transactions.account_no
group by player_id, transactions.account_no
order by player_id asc;
```



No. of Tuples : 114

6. **List all the transactions in which the player gets rewarded the amount in Cash.**

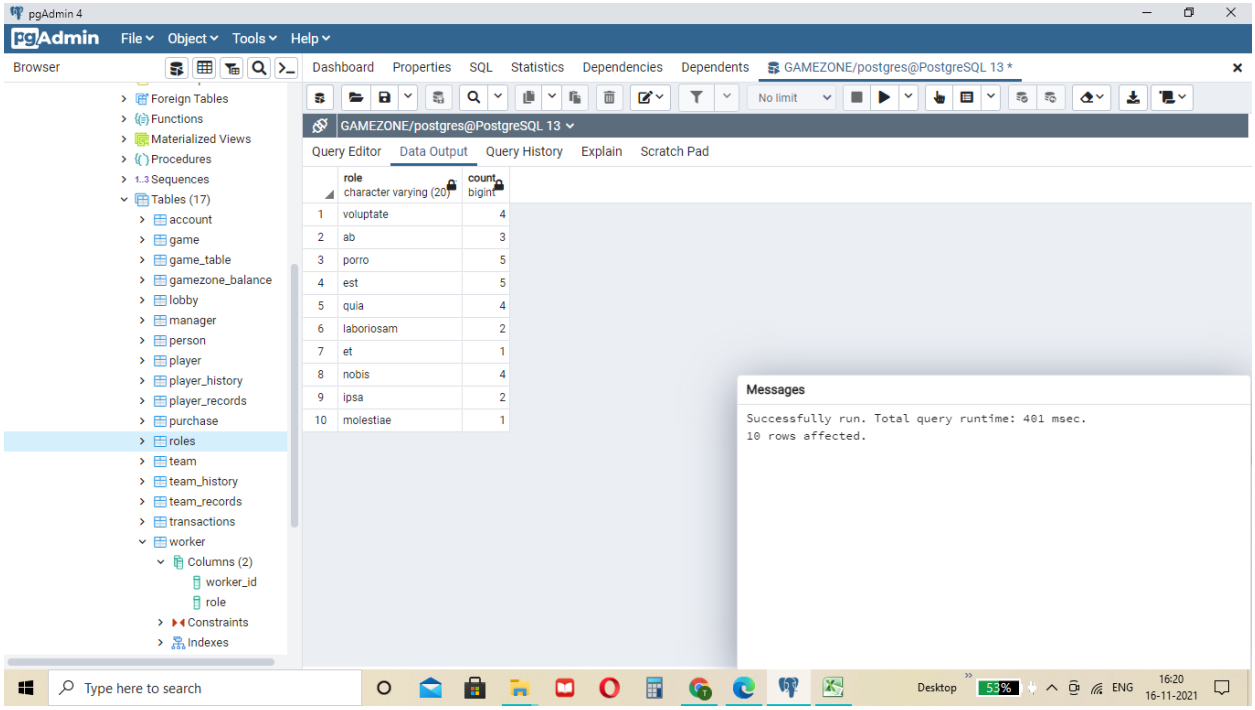
```
select * from transactions
where type = 'win' and method = 'Cash';
```



No of tuple = 35

7. Count the number of workers for each role.

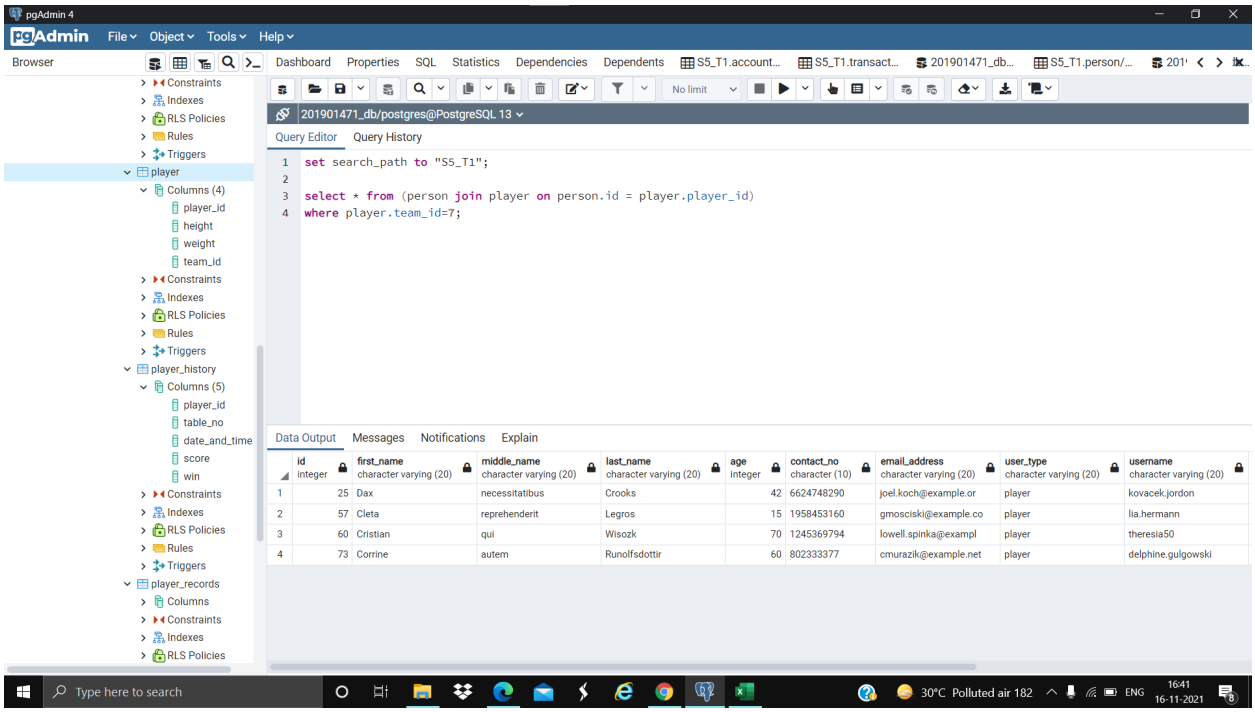
```
SELECT role, COUNT(*)
FROM worker
GROUP BY role;
```



No of tuple = 10

8. Find details of players who have team_id 7.

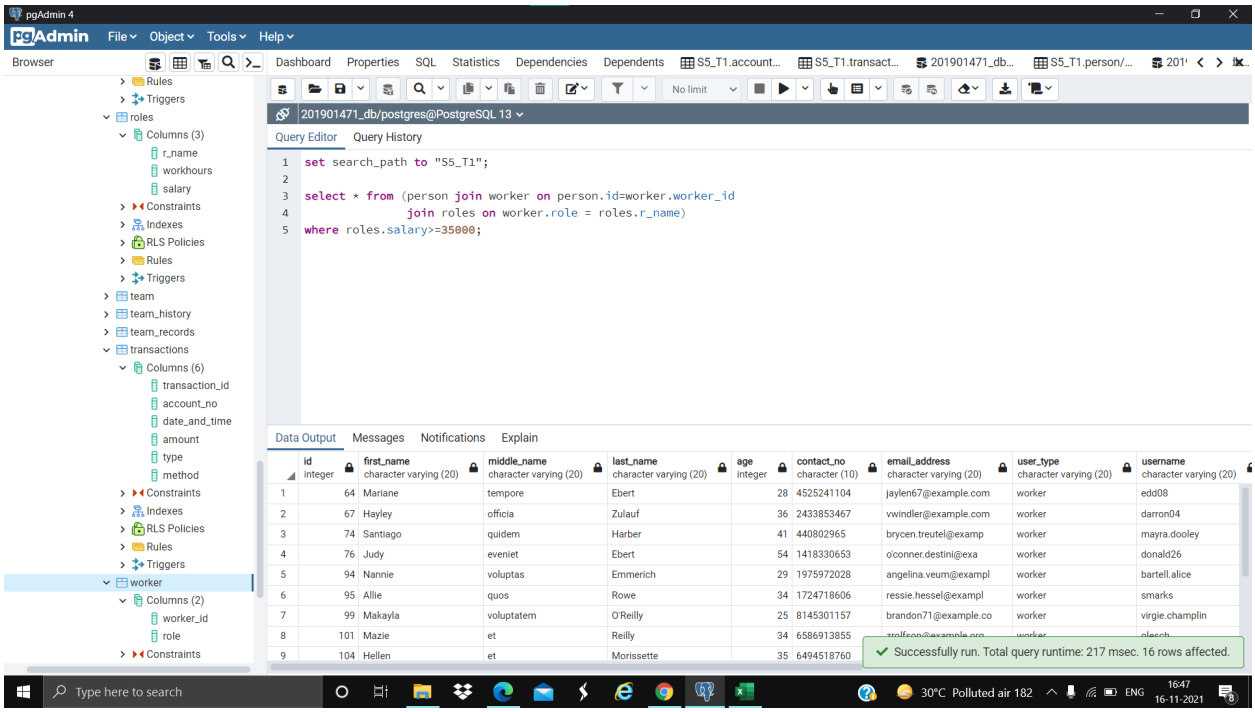
```
SELECT * FROM
        (person JOIN player ON person.id = player.player_id)
    WHERE player.team_id = 7
```



No. of tuple : 4

9. Find details of workers who have more than 35000 salary.

```
select * from (person join worker on person.id=worker.worker_id
                    join roles on worker.role = roles.r_name)
where roles.salary>=35000;
```

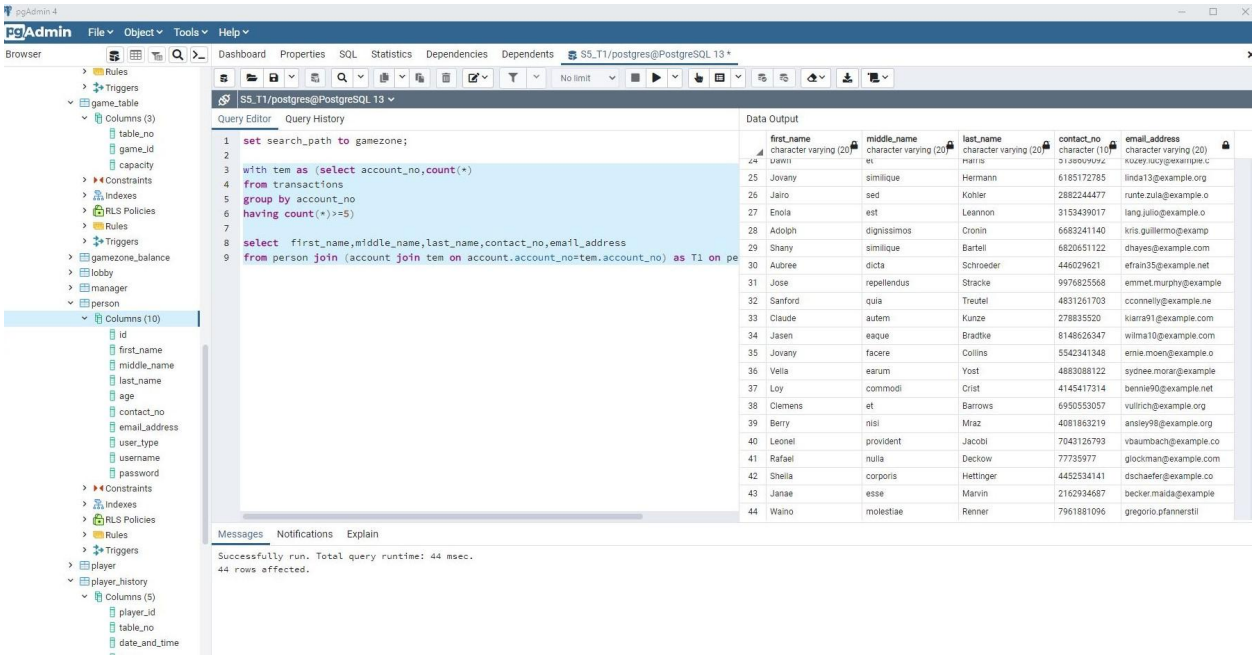


	id integer	first_name character varying (20)	middle_name character varying (20)	last_name character varying (20)	age integer	contact_no character (10)	email_address character varying (20)	user_type character varying (20)	username character varying (20)
1	64	Mariane	tempore	Ebert	28	4525241104	jaylen67@example.com	worker	edd08
2	67	Hayley	officia	Zulauf	36	2433853467	vwindler@example.com	worker	darron04
3	74	Santiago	quidem	Harber	41	440802965	brycen.treutel@examp	worker	mayra.dooley
4	76	Judy	eveniet	Ebert	54	1418330653	o'conner.destini@exa	worker	donald26
5	94	Nannie	voluptas	Emmerich	29	1975972028	angelina.veum@exampl	worker	bartell.alice
6	95	Allie	quos	Rowe	34	1724718606	ressie.hessel@exampl	worker	smarks
7	99	Makayla	voluptatem	O'Reilly	25	8145301157	brandon71@example.co	worker	virgie.champlin
8	101	Mazie	et	Reilly	34	6586913855	[illegible]	worker	[illegible]
9	104	Hellen	et	Morissette	35	6494518760			

No. of tuple = 16

10. Find the person details who have more than 3 transactions.

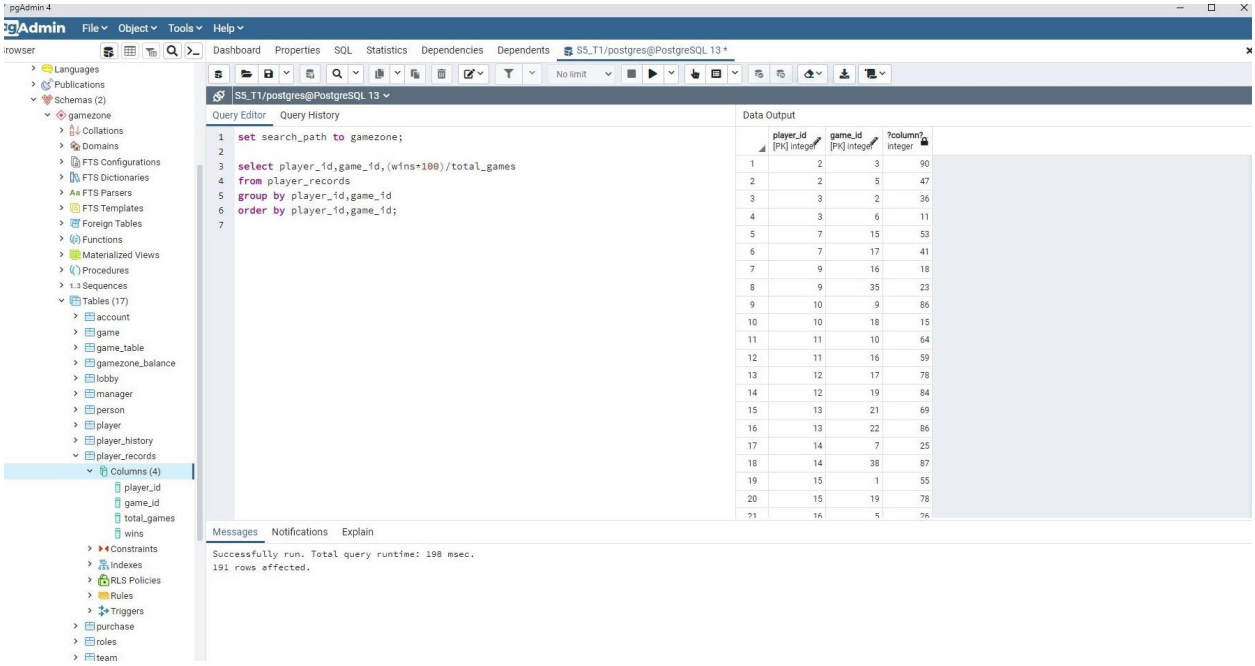
```
with tem as (select account_no,count(*)
from transactions
group by account_no
having count(*)>=5)

select  first_name,middle_name,last_name,contact_no,email_address
from person join (account join tem on
account.account_no=tem.account_no) as T1 on person.id=T1.player_id;
```



No. of tuple = 44

11. Find the win rate of each player gamewise.

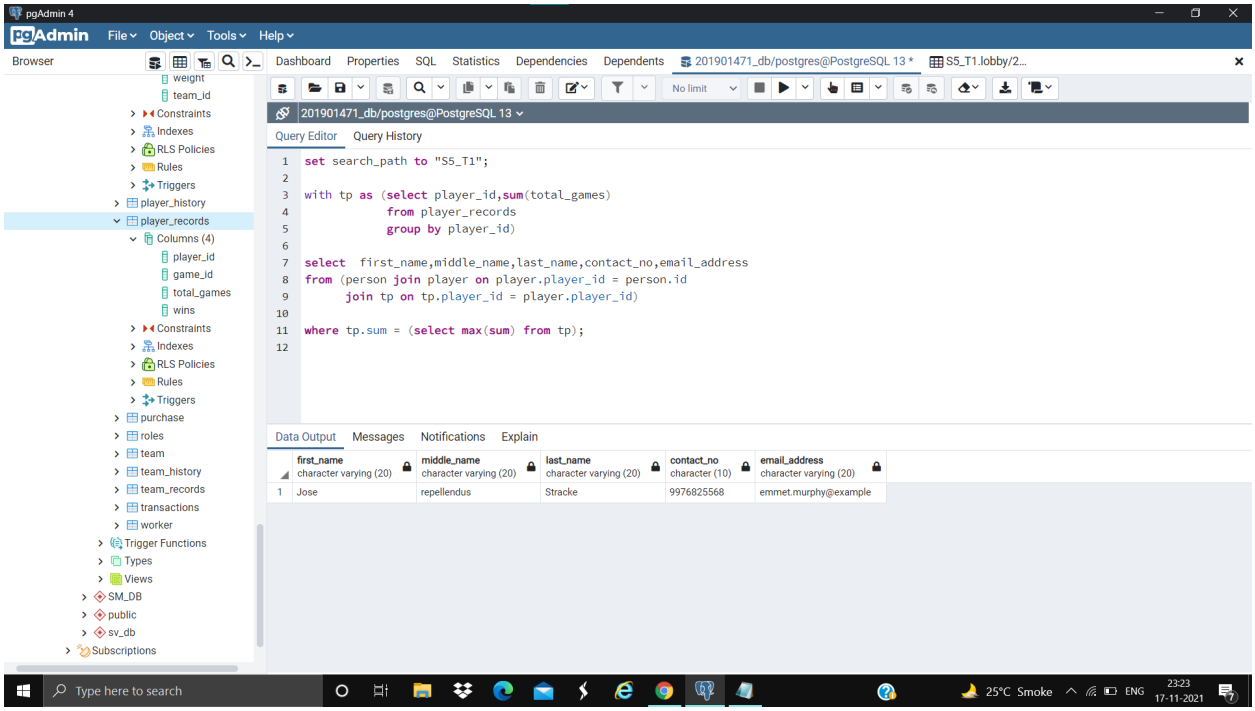
```
select player_id,game_id,(wins*100)/total_games
from player_records
group by player_id,game_id
order by player_id,game_id;
```



No. of tuples = 191.

12. Find the player detail who played most games.

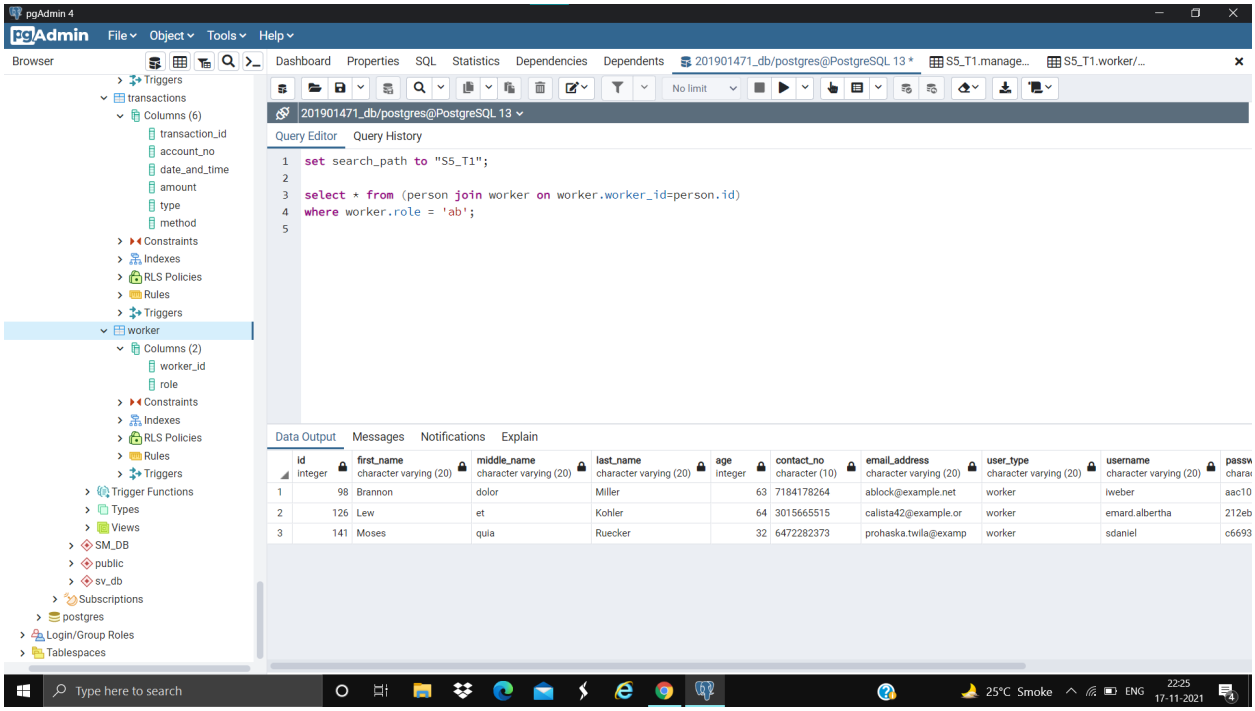
```
with tp as (select player_id,sum(total_games)
                    from player_records
                    group by player_id)

select  first_name,middle_name,last_name,contact_no,email_address
from (person join player on player.player_id = person.id
        join tp on tp.player_id = player.player_id)

where tp.sum = (select max(sum) from tp);
```



No. of tuple : 1

13. Find the Worker list who has the role “ab”.

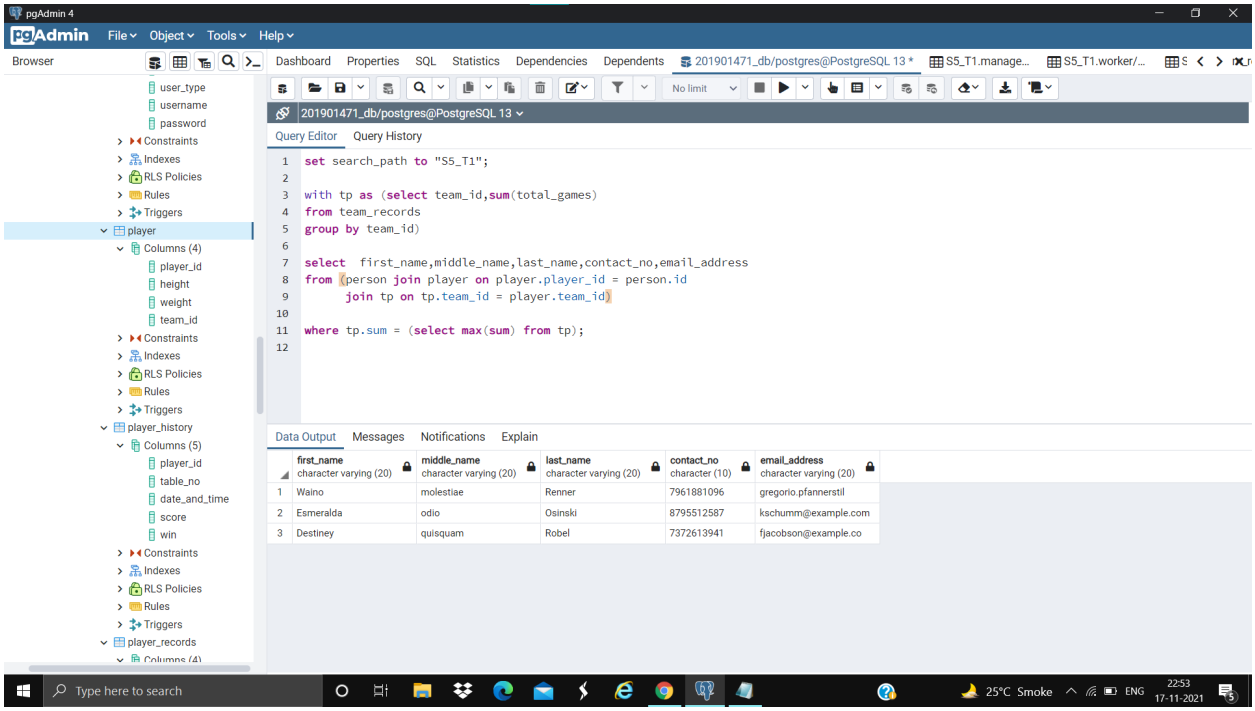
```
select * from (person join worker on worker.worker_id=person.id)
where worker.role = 'ab';
```



no of tuple = 3

14. List the team players who played most games in teams.

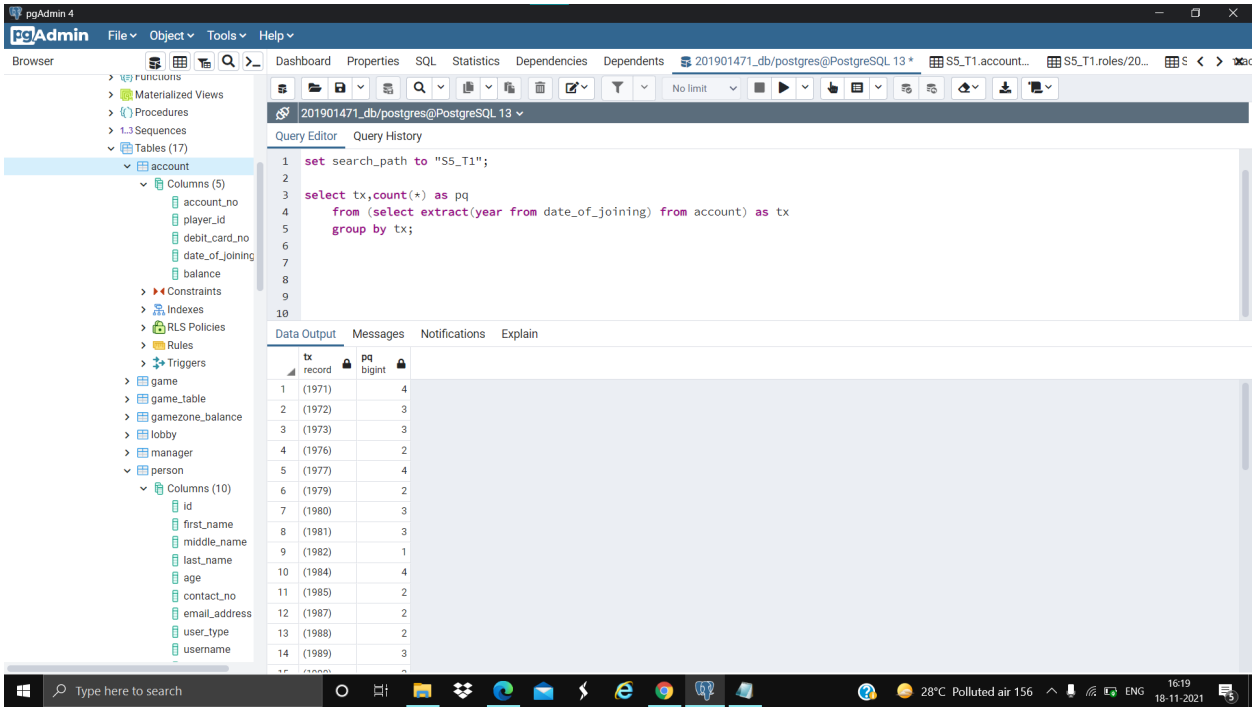
```
with tp as (select team_id,sum(total_games)
from team_records
group by team_id)

select  first_name,middle_name,last_name,contact_no,email_address
from (person join player on player.player_id = person.id
        join tp on tp.team_id = player.team_id)

where tp.sum = (select max(sum) from tp);
```



No. of tuple : 3

15. Show data how many players joined each year.

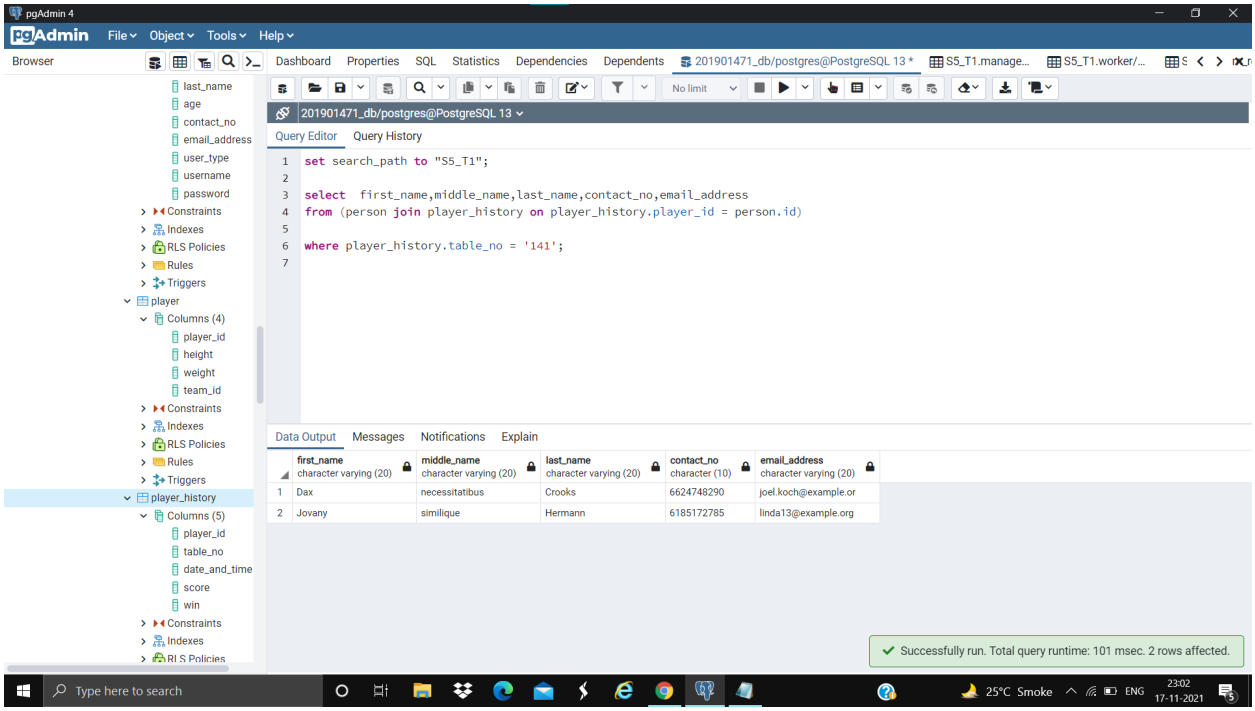
```
select tx,count(*) as pq
        from (select extract(year from date_of_joining) from account) as
tx
        group by tx;
```



No. of tuple : 45

16. List all players who played on table number 141.

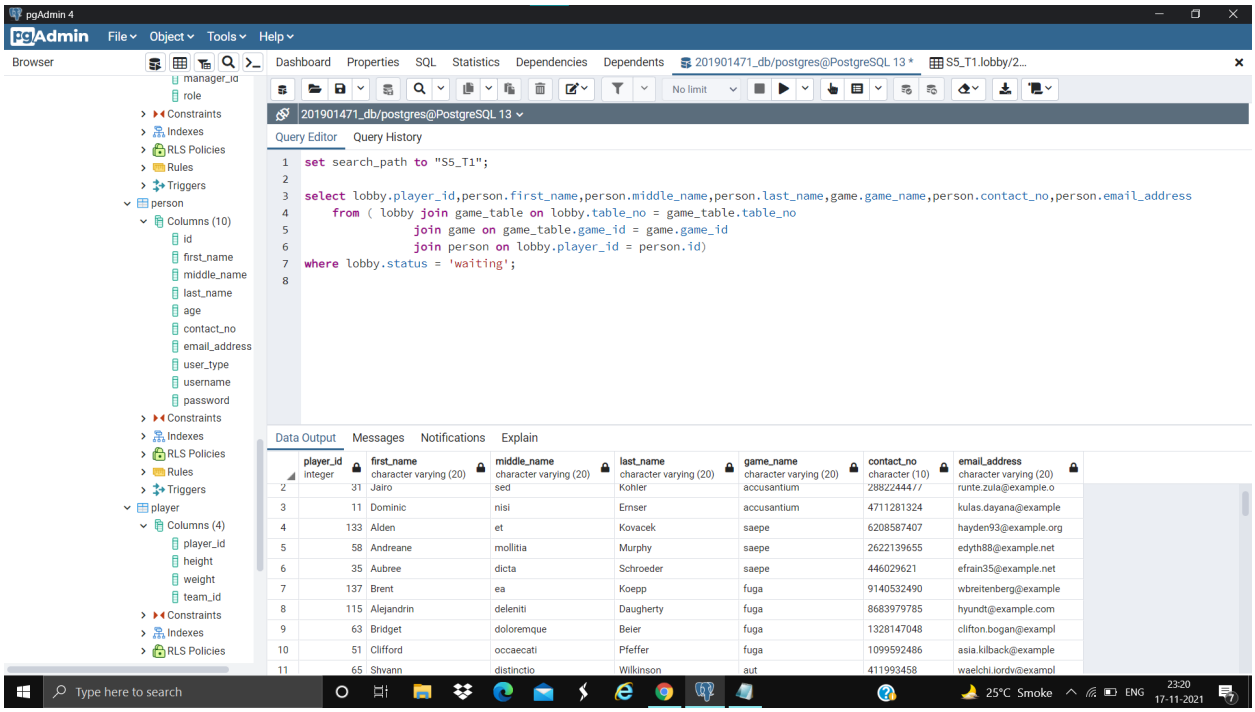
```
select  first_name,middle_name,last_name,contact_no,email_address
from (person join player_history on player_history.player_id = person.id)

where player_history.table_no = '141';
```



No. of tuple : 2

17. Find the list of players who are waiting in the lobby and also show the game name for which they are waiting.

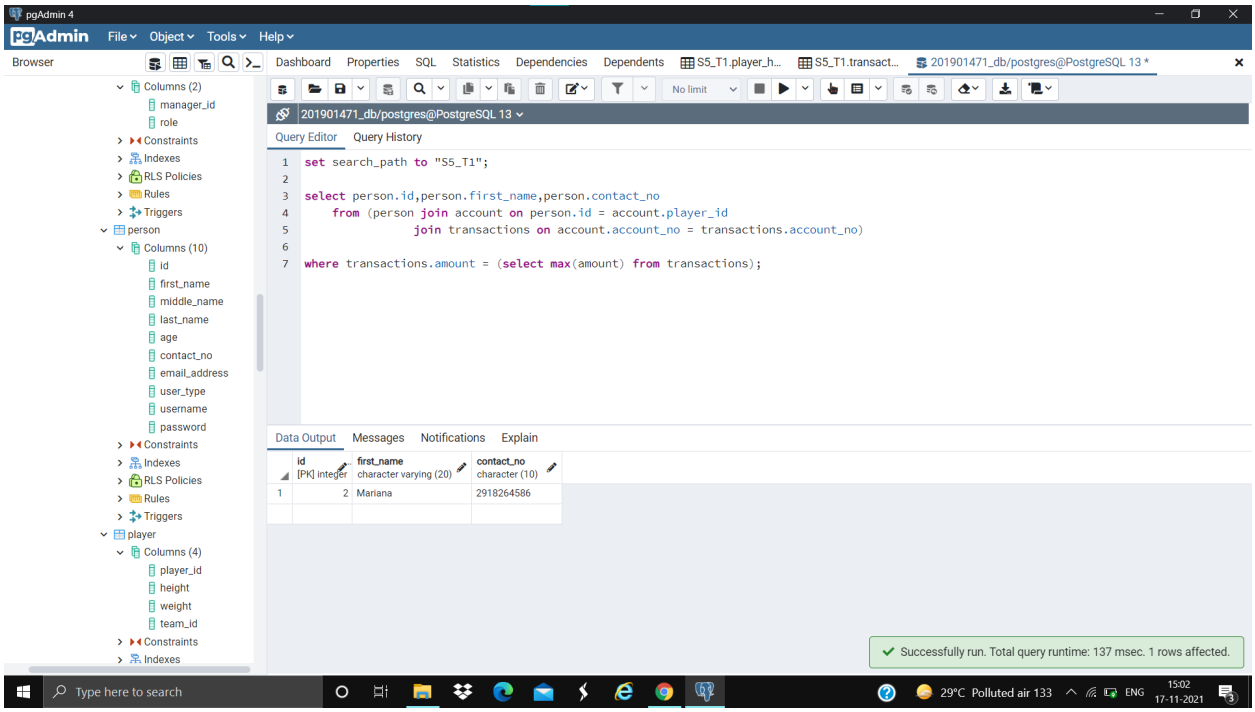
```
select
lobby.player_id,person.first_name,person.middle_name,person.last_name,
game.game_name,person.contact_no,person.email_address
        from ( lobby join game_table on lobby.table_no =
game_table.table_no
                        join game on game_table.game_id =
game.game_id
                        join person on lobby.player_id = person.id)
where lobby.status = 'waiting';
```



No. of tuple : 57

18. Find the person id, contact number and name who has the biggest transaction amount.

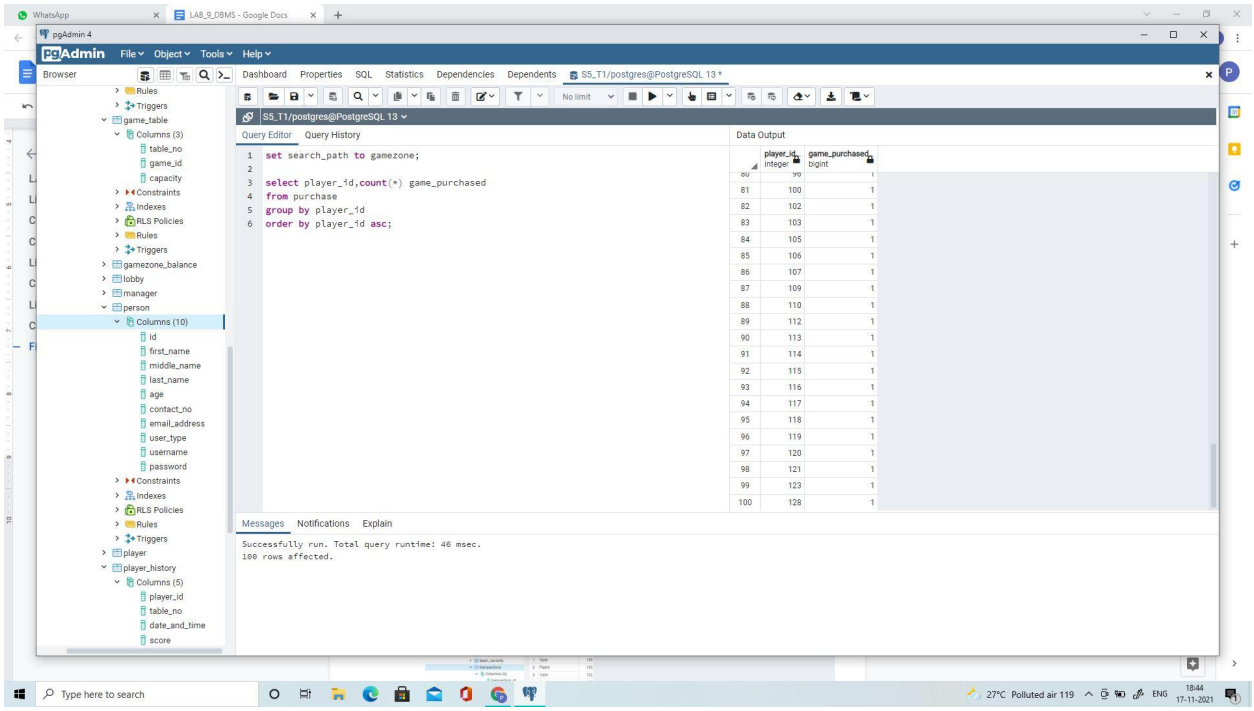
```
select person.id,person.first_name,person.contact_no
        from (person join account on person.id = account.player_id
                        join transactions on account.account_no =
transactions.account_no)

where transactions.amount = (select max(amount) from transactions);
```



No of tuple = 1

19. Number of game purchased by each player.

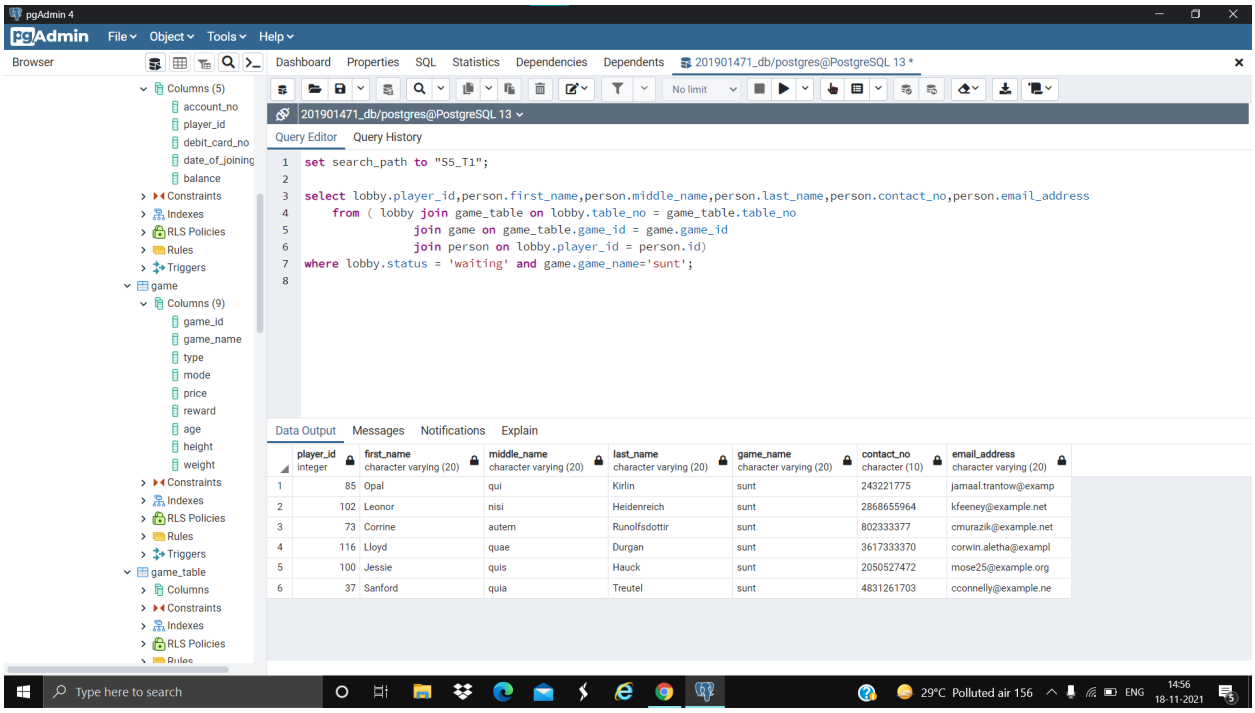
```
select player_id,count(*) game_purchased
from purchase
group by player_id
order by player_id asc;
```



No. of tuple=100

20. Find the player details who are waiting for the game "sunt".

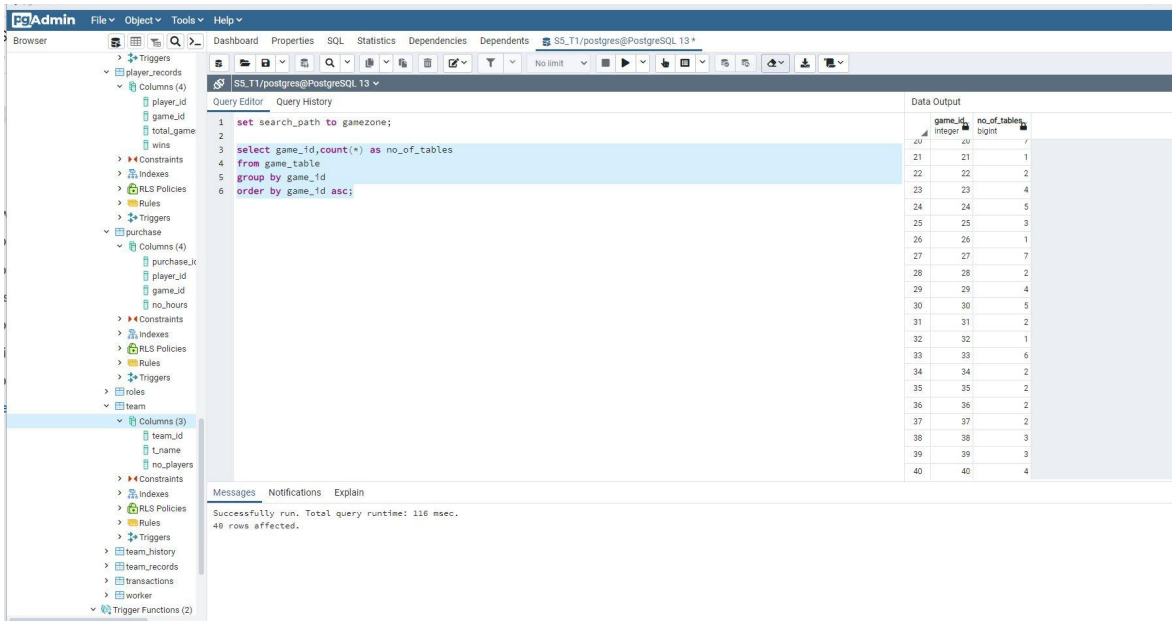
```
select
lobby.player_id,person.first_name,person.middle_name,person.last_name,
person.contact_no,person.email_address

        from ( lobby join game_table on lobby.table_no =
game_table.table_no

                        join game on game_table.game_id =
game.game_id

                        join person on lobby.player_id = person.id)

where lobby.status = 'waiting' and game.game_name='sunt';
```



No. of tuple : 6

21. Find the number of tables for each game.

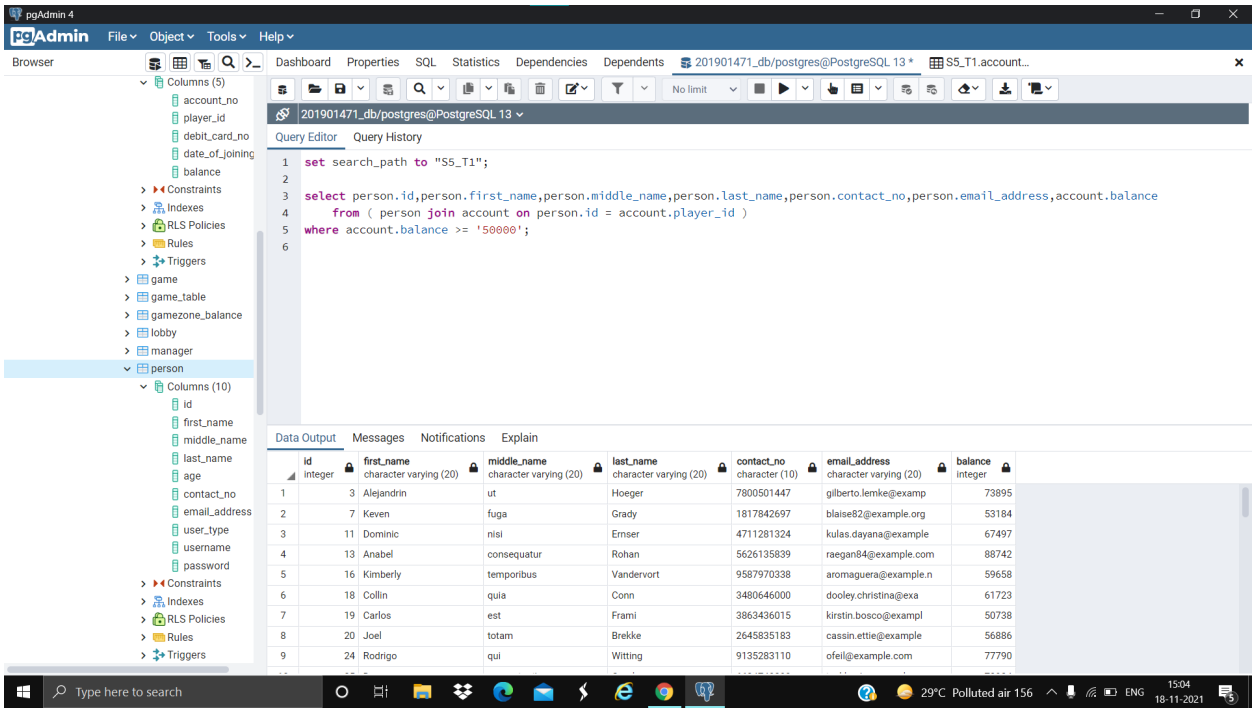
```
select game_id,count(*) as no_of_tables
from game_table
group by game_id
order by game_id asc;
```



No. of tuple : 40

22.Find the player details who has balance more than “50000”.

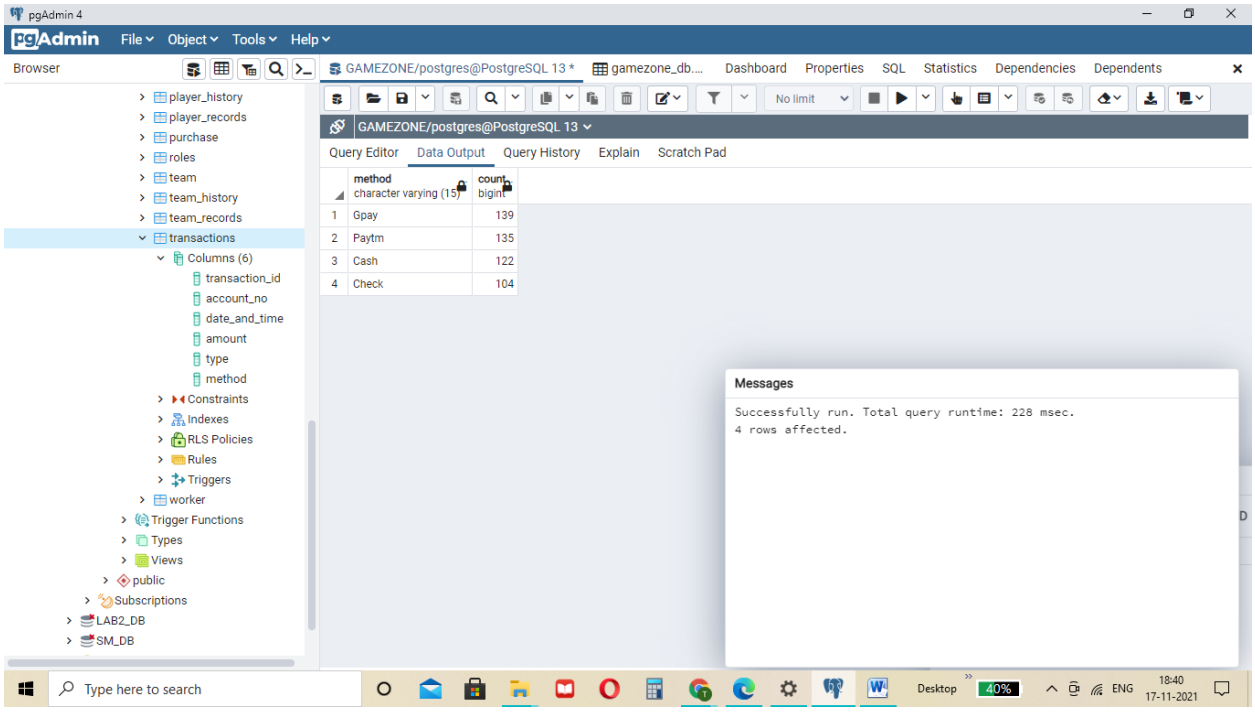
```
select
person.id,person.first_name,person.middle_name,person.last_name,perso
n.contact_no,person.email_address,account.balance

        from ( person join account on person.id = account.player_id )

where account.balance >= '50000';
```



No. of tuple : 49

23.Find the number of transactions done by each method.

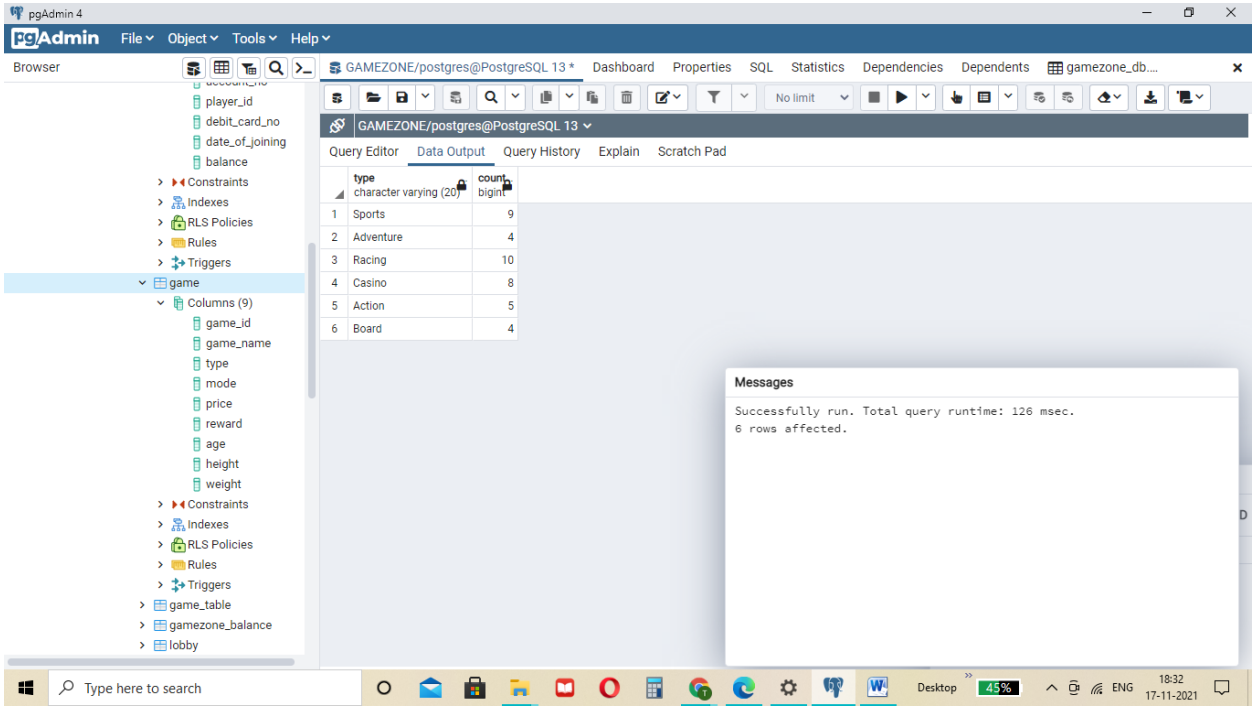
```
SELECT transactions.method, COUNT(*)
FROM transactions
GROUP BY transactions.method
```



No of tuple = 4

24. Find the number of games for each type.

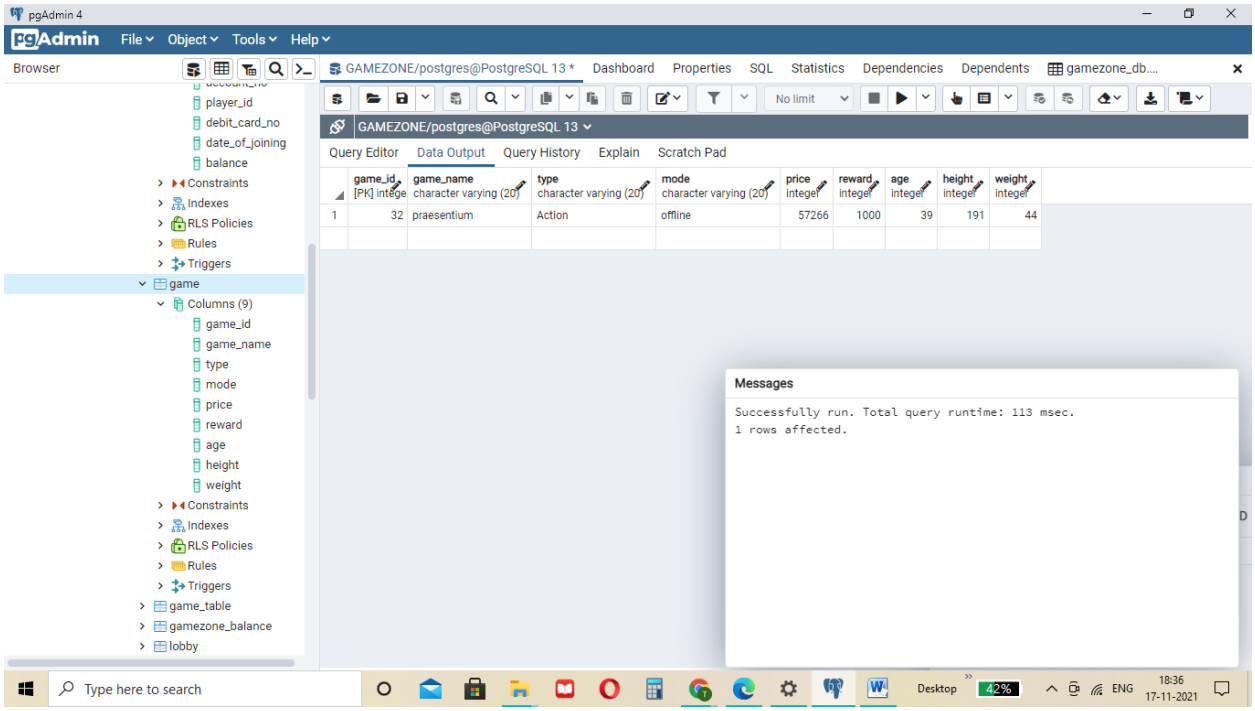
```
SELECT game.type, count(*)
FROM game GROUP BY game.type;
```



No of tuple = 6

25.Find the game details which have the highest reward.

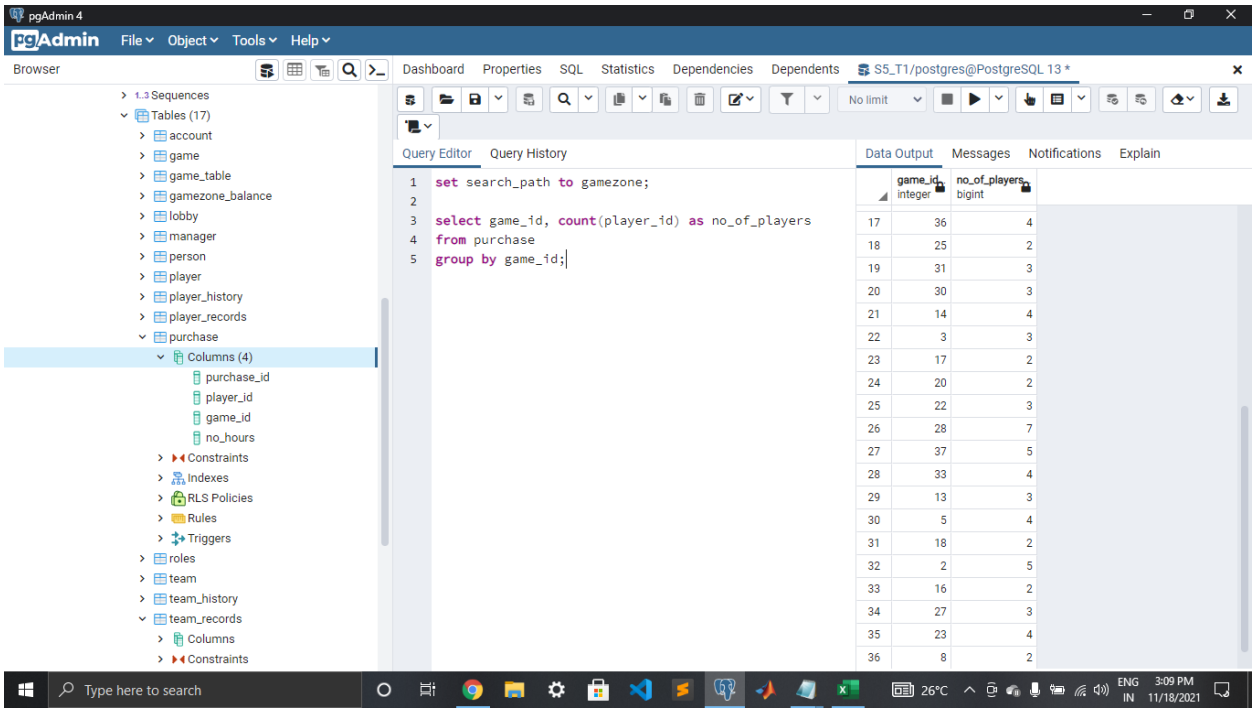
```
SELECT * FROM game
WHERE game.reward =
        (SELECT MAX(game.reward)
        FROM game)
```



No of tuple = 1

26. Number of players purchase each game.

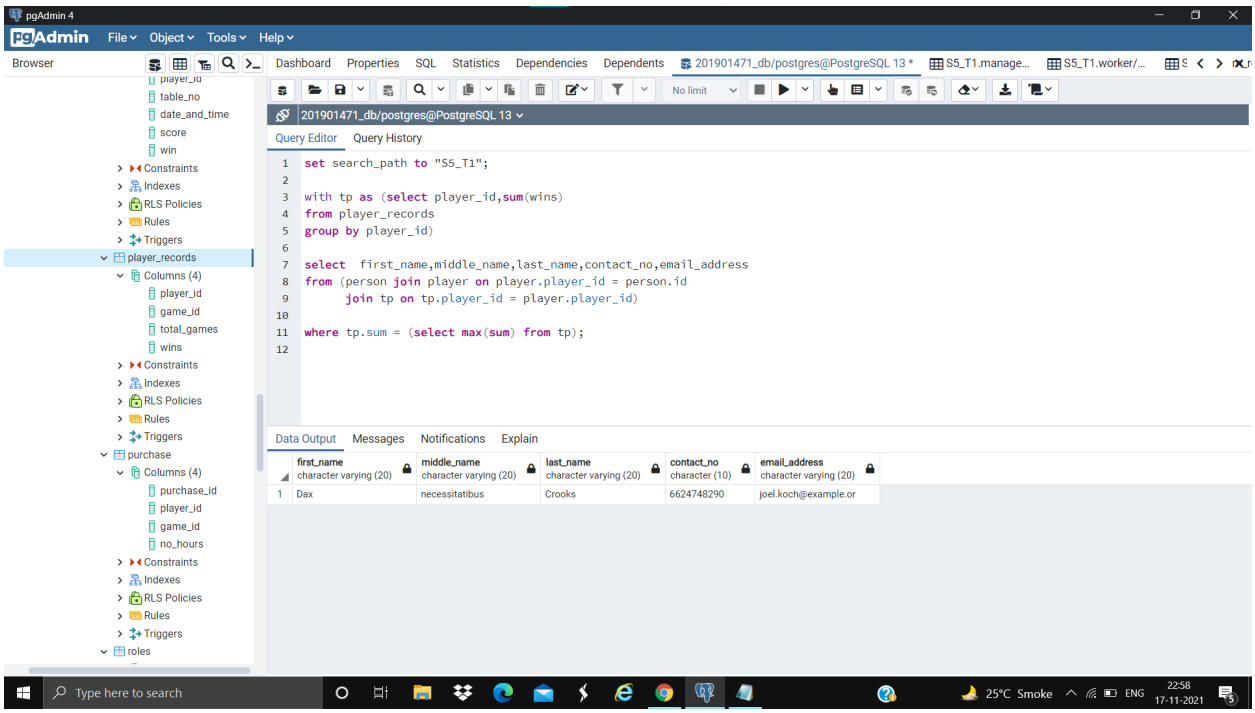
```
select game_id, count(player_id) as no_of_players
from purchase
group by game_id;
```



No. of Tuple : 37

27. Find the player who wins most times.

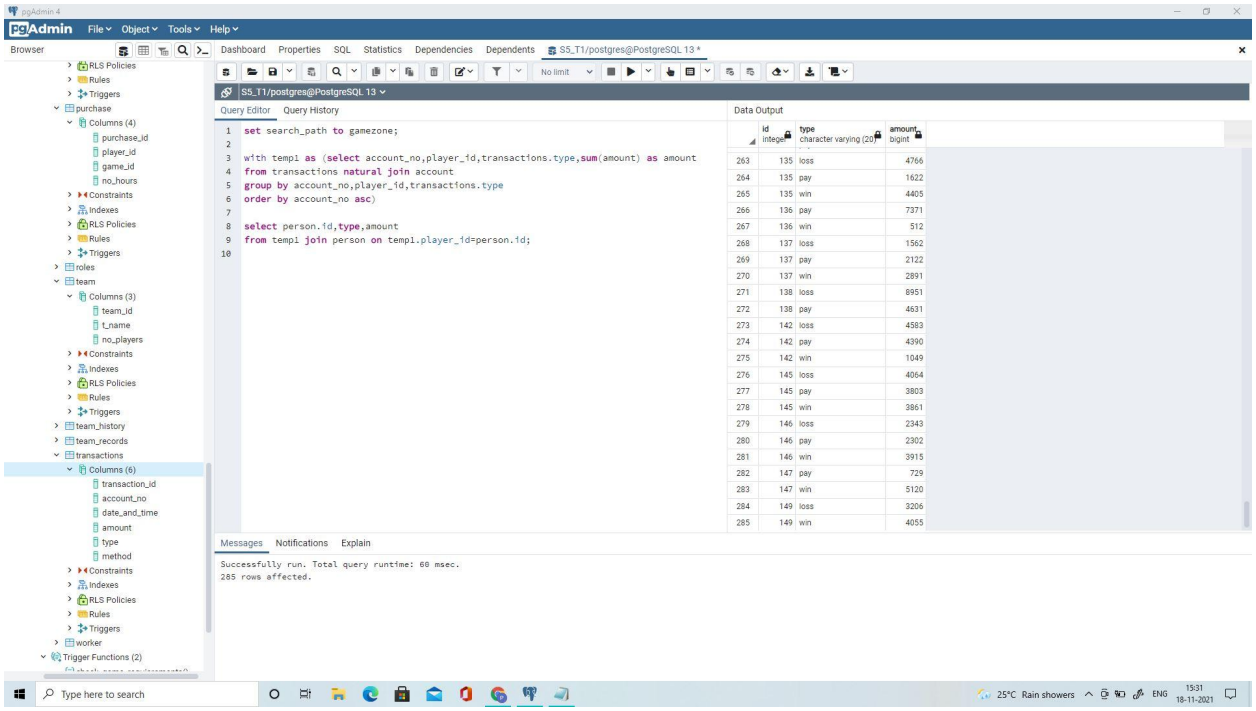
```
with tp as (select player_id,sum(wins)
from player_records
group by player_id)

select  first_name,middle_name,last_name,contact_no,email_address
from (person join player on player.player_id = person.id
        join tp on tp.player_id = player.player_id)

where tp.sum = (select max(sum) from tp);
```



No. of Tuple : 1

28. List the total reward, loss and pay amount of each player.

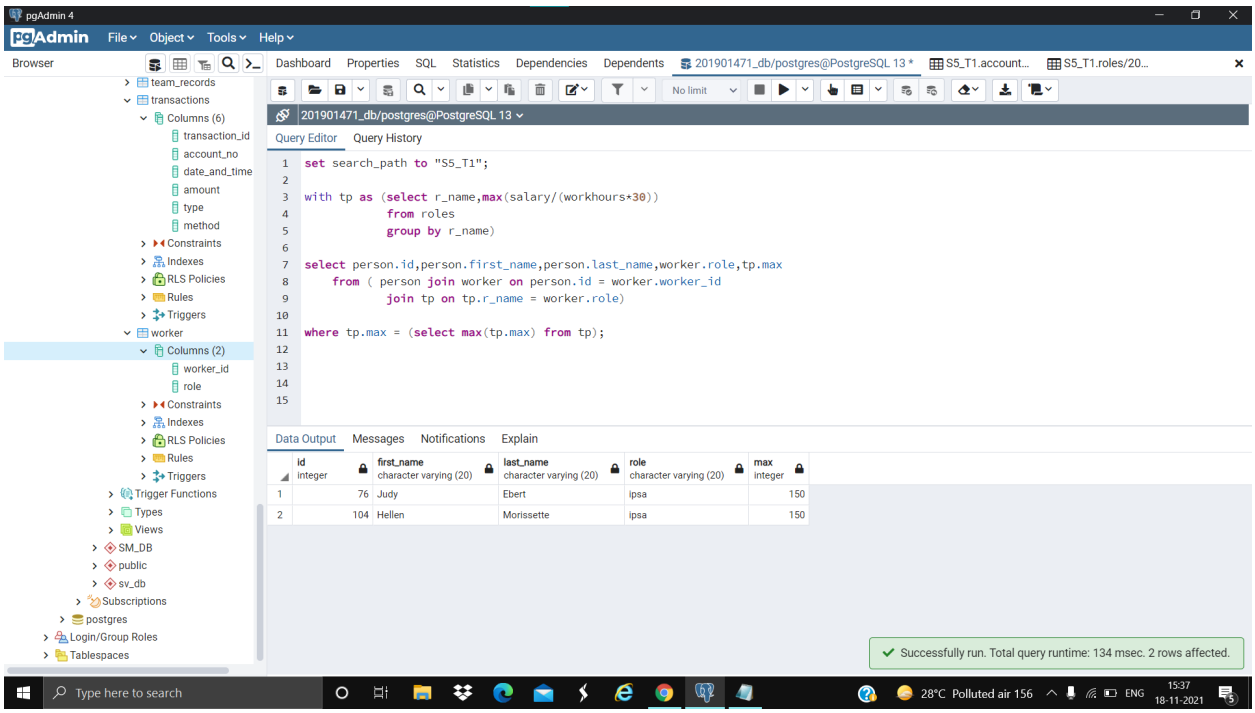
```
with temp1 as (select
account_no,player_id,transactions.type,sum(amount) as amount
from transactions natural join account
group by account_no,player_id,transactions.type
order by account_no asc)

select person.id,type,amount
from temp1 join person on temp1.player_id=person.id;
```



No. of tuple = 285

29.Find the details of the worker who has the highest salary for 1 hour.

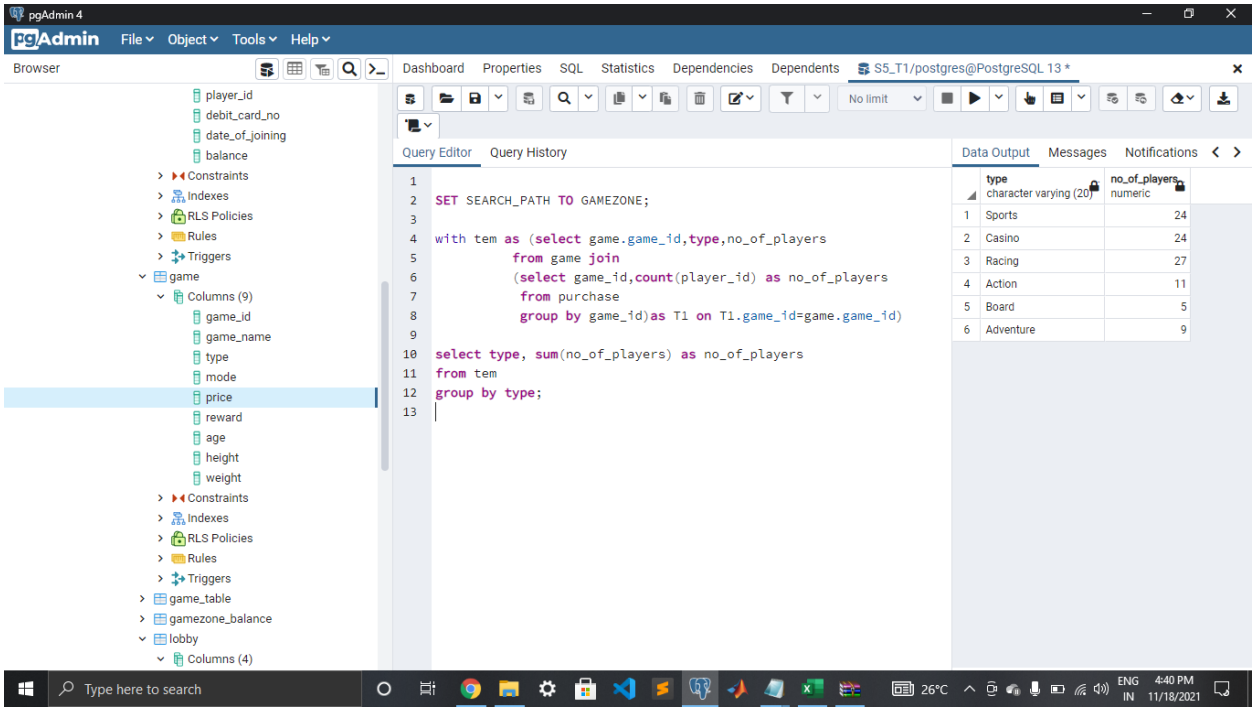
```
with tp as (select r_name,max(salary/(workhours*30))
                    from roles
                    group by r_name)

select person.id,person.first_name,person.last_name,worker.role,tp.max
    from ( person join worker on person.id = worker.worker_id
                    join tp on tp.r_name = worker.role)

where tp.max = (select max(tp.max) from tp);
```



No. of tuple : 2;

30. Which type of game is played most?

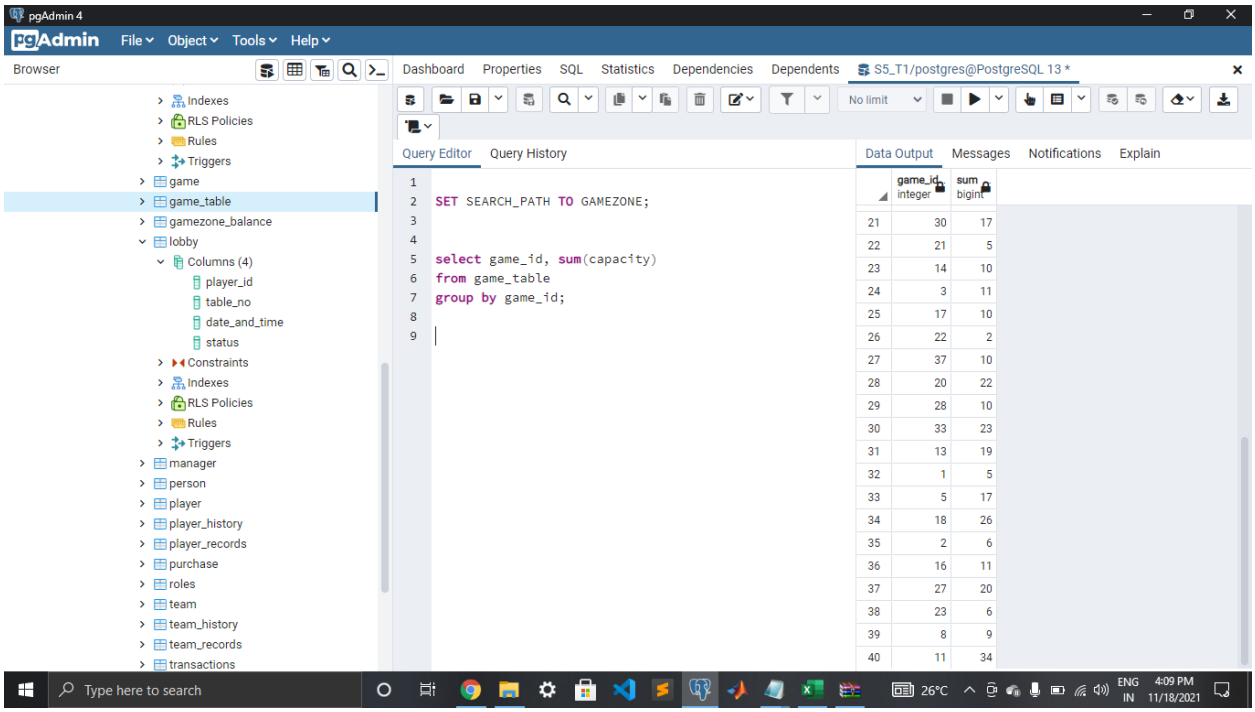
```
with tem as (select game.game_id,type,no_of_players
        from game join
        (select game_id,count(player_id) as no_of_players
         from purchase
         group by game_id)as T1 on T1.game_id=game.game_id)

select type, sum(no_of_players) as no_of_players
from tem
group by type;
```



No. of Tuple : 6

31. Find the number of players can play particular game at same time

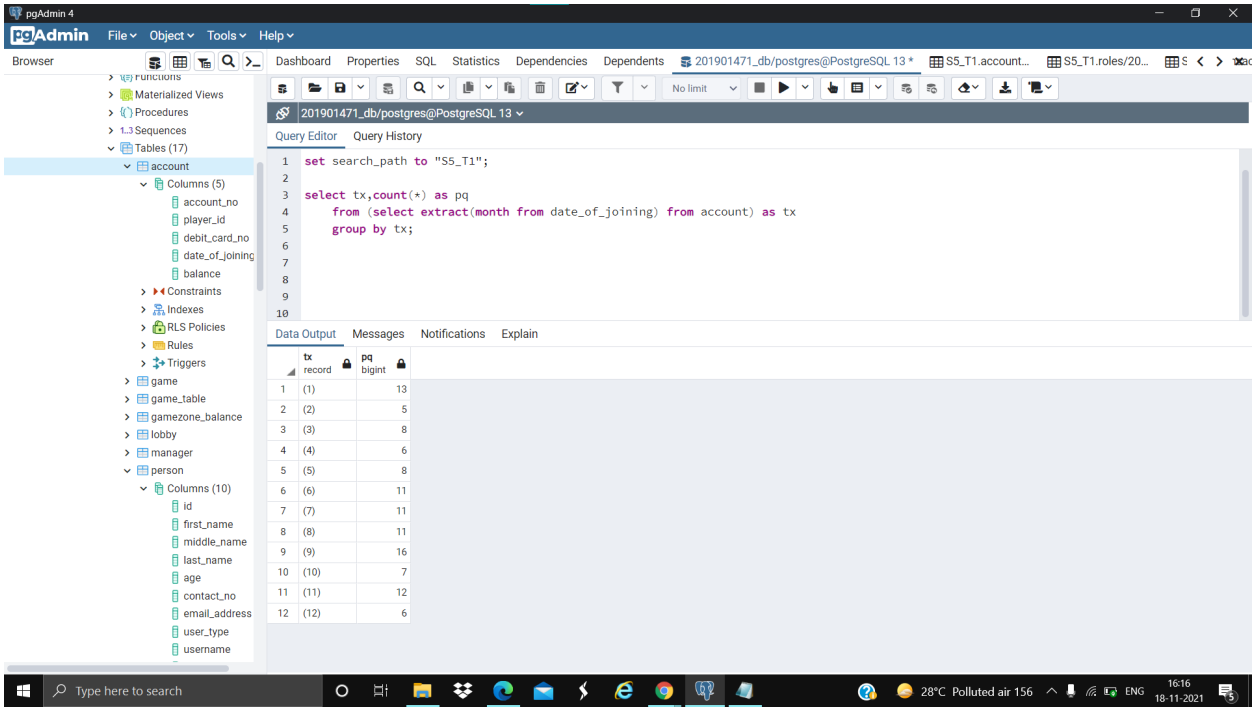
```
select game_id, sum(capacity)
from game_table
group by game_id;
```



No. of Tuple : 40

32. Analyze the data and list the total number of players joined based on months.

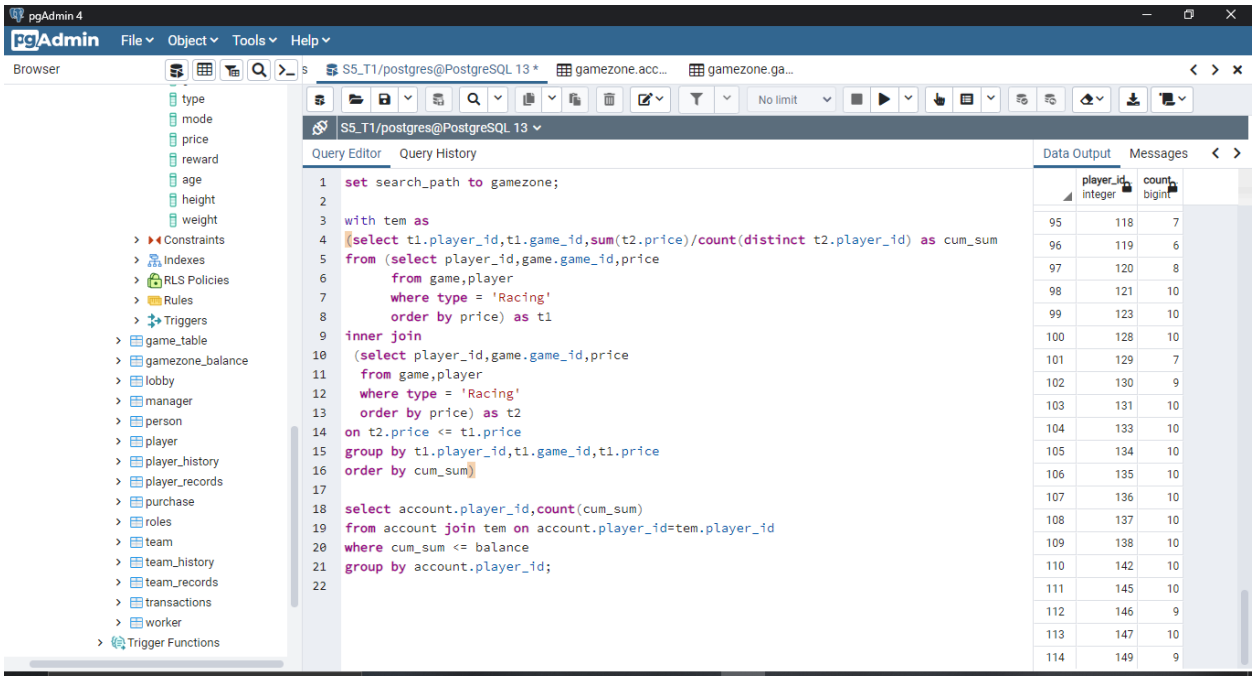
```
select tx, count(*) as pq
        from (select extract(month from date_of_joining) from account) as
tx
        group by tx;
```



No. of tuple : 12

33. Count the number of different games of ‘Racing’ type a player can purchase from his/her balance.

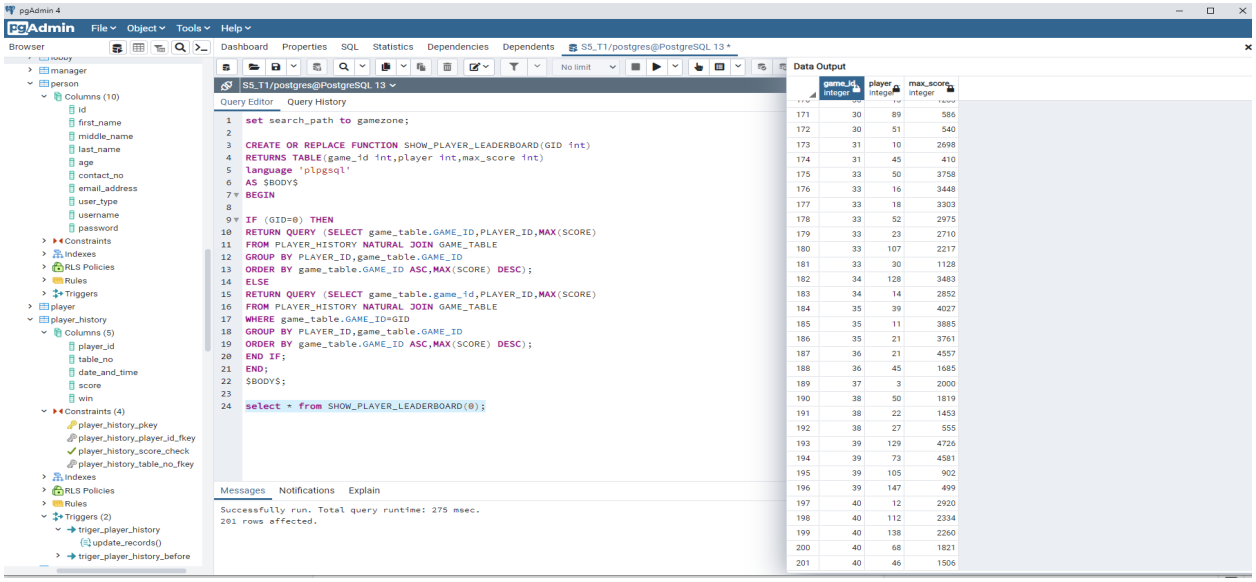
```
with tem as (select t1.player_id,t1.game_id,sum(t2.price)/count(distinct
t2.player_id) as cum_sum
from (select player_id,game.game_id,price from game,player where type
= 'Racing' order by price) as t1
inner join (select player_id,game.game_id,price from game,player where
type = 'Racing' order by price) as t2 on t2.price <= t1.price
group by t1.player_id,t1.game_id,t1.price
order by cum_sum)

select account.player_id,count(cum_sum) from account join tem on
account.player_id=tem.player_id
where cum_sum <= balance
group by account.player_id;
```



No. of tuple : 114

34. Show Leaderboard of all games.

select * from SHOW_PLAYER_LEADERBOARD(0);

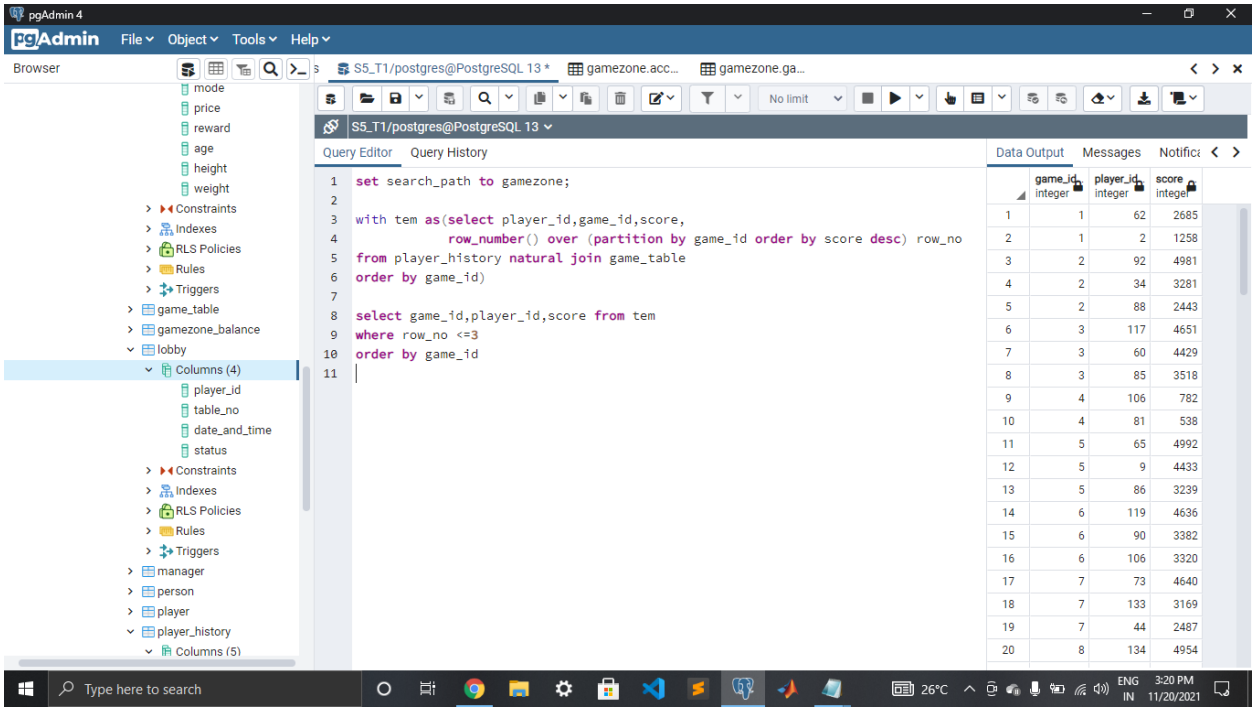



```
set search_path to gamezone;

CREATE OR REPLACE FUNCTION SHOW_PLAYER_LEADERBOARD(GID int)
RETURNS TABLE(game_id int,player int,max_score int)
language 'plpgsql'
AS $BODY$
BEGIN

IF (GID=0) THEN
RETURN QUERY (SELECT game_table.GAME_ID,PLAYER_ID,MAX(SCORE)
FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
GROUP BY PLAYER_ID,game_table.GAME_ID
ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
ELSE
RETURN QUERY (SELECT game_table.game_id,PLAYER_ID,MAX(SCORE)
FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
WHERE game_table.GAME_ID=GID
GROUP BY PLAYER_ID,game_table.GAME_ID
ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
END IF;
END;
$BODY$;

select * from SHOW_PLAYER_LEADERBOARD(0);
```

No. of tuple = 201;

35.List the top 3 scoring players of each game.

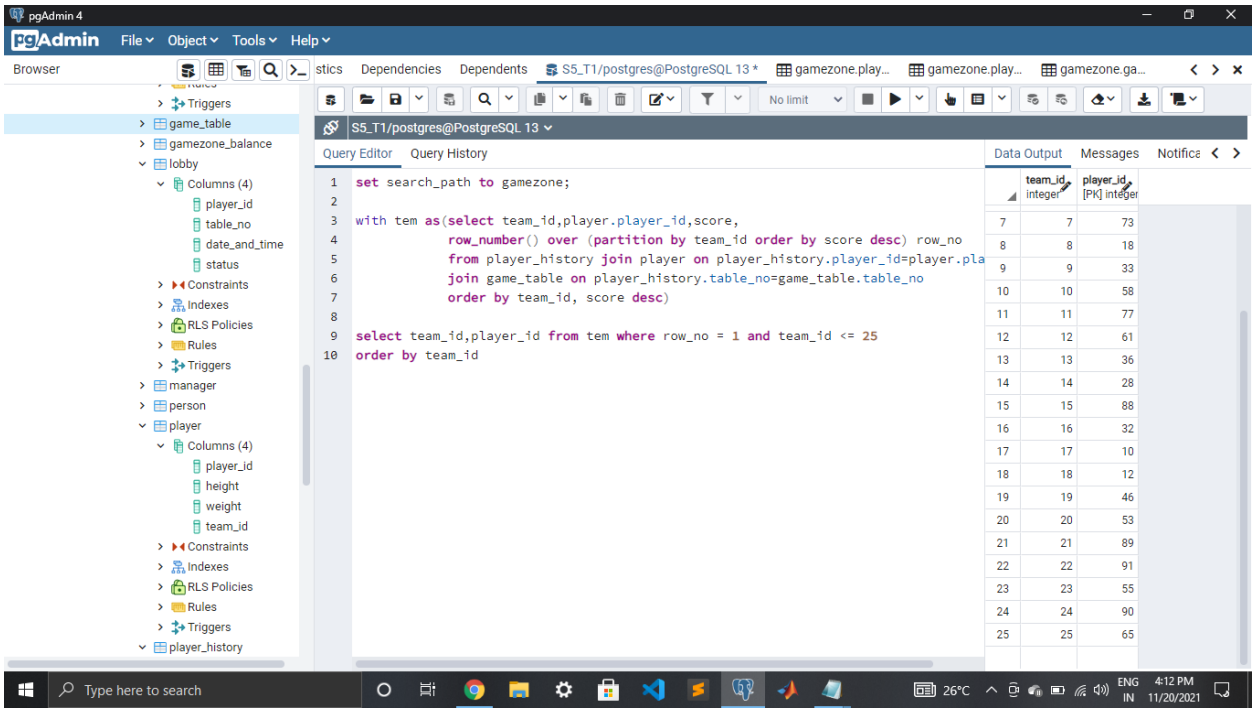
```
set search_path to gamezone;
with tem as(select player_id,game_id,score,
                    row_number() over (partition by game_id order by
score desc) row_no
from player_history natural join game_table
order by game_id)

select game_id,player_id,score from tem
where row_no <=3
order by game_id
```



No of tuple = 100

36.List the best player in each team according to higher scores.

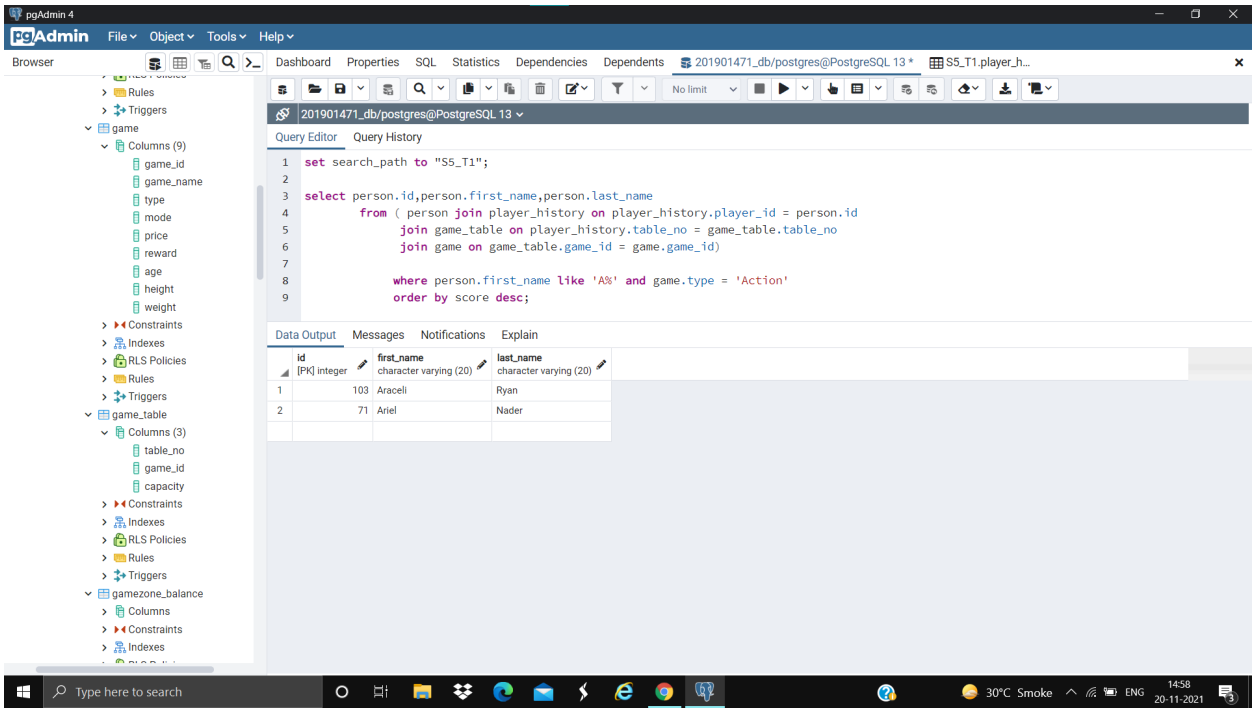
```
with tem as(select team_id,player.player_id,score,
                    row_number() over (partition by team_id order by
score desc) row_no
                    from player_history join player on
player_history.player_id=player.player_id
                    join game_table on
player_history.table_no=game_table.table_no
                    order by team_id, score desc)

select team_id,player_id from tem where row_no = 1 and team_id <= 25
order by team_id
```



No. of tuple : 25

37. List the player id and name whose name starts with letter ‘A’ and order them descending order of their score in Action games.

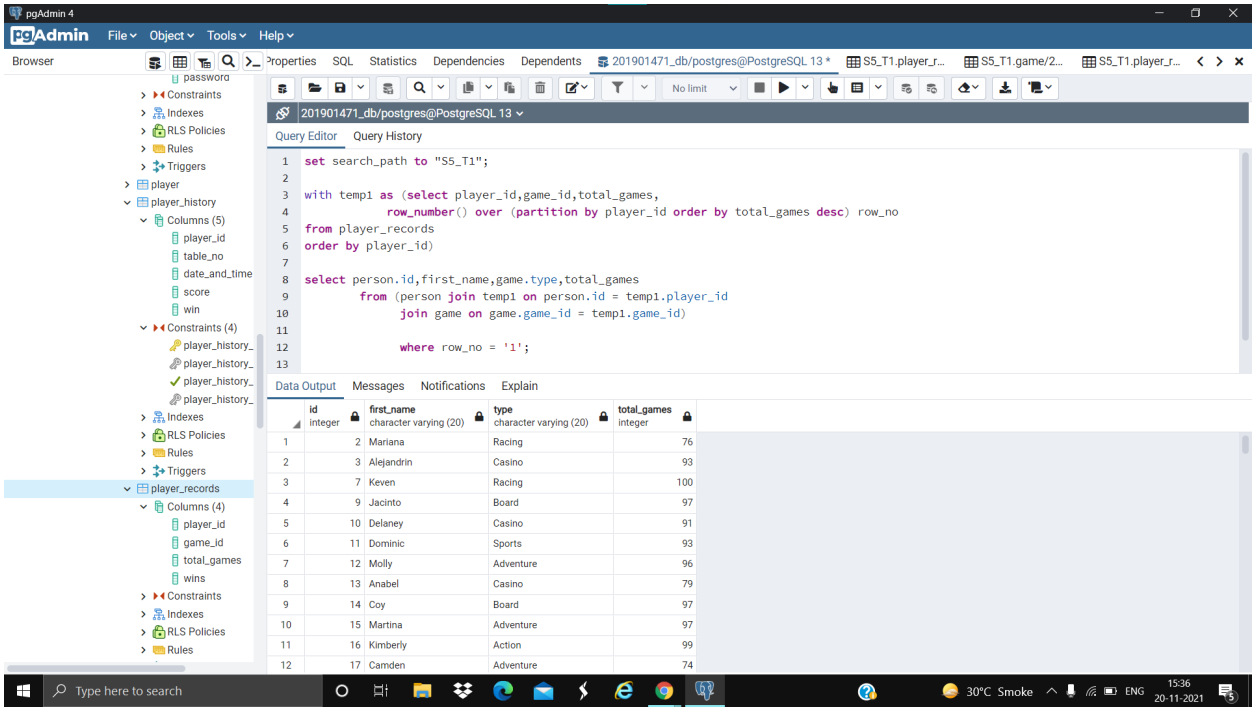
```
select person.id,person.first_name,person.last_name
                from ( person join player_history on player_history.player_id
= person.id
                        join game_table on player_history.table_no =
game_table.table_no
                        join game on game_table.game_id = game.game_id)

                       where person.first_name like 'A%' and game.type =
'Action'
                       order by score desc;
```



No. of tuple : 2

38. List the player details and which type of game he played most.

```
with temp1 as (select player_id,game_id,total_games,
                    row_number() over (partition by player_id order by
total_games desc) row_no
from player_records
order by player_id)

select person.id,first_name,game.type,total_games
        from (person join temp1 on person.id = temp1.player_id
              join game on game.game_id = temp1.game_id)

              where row_no = '1';
```

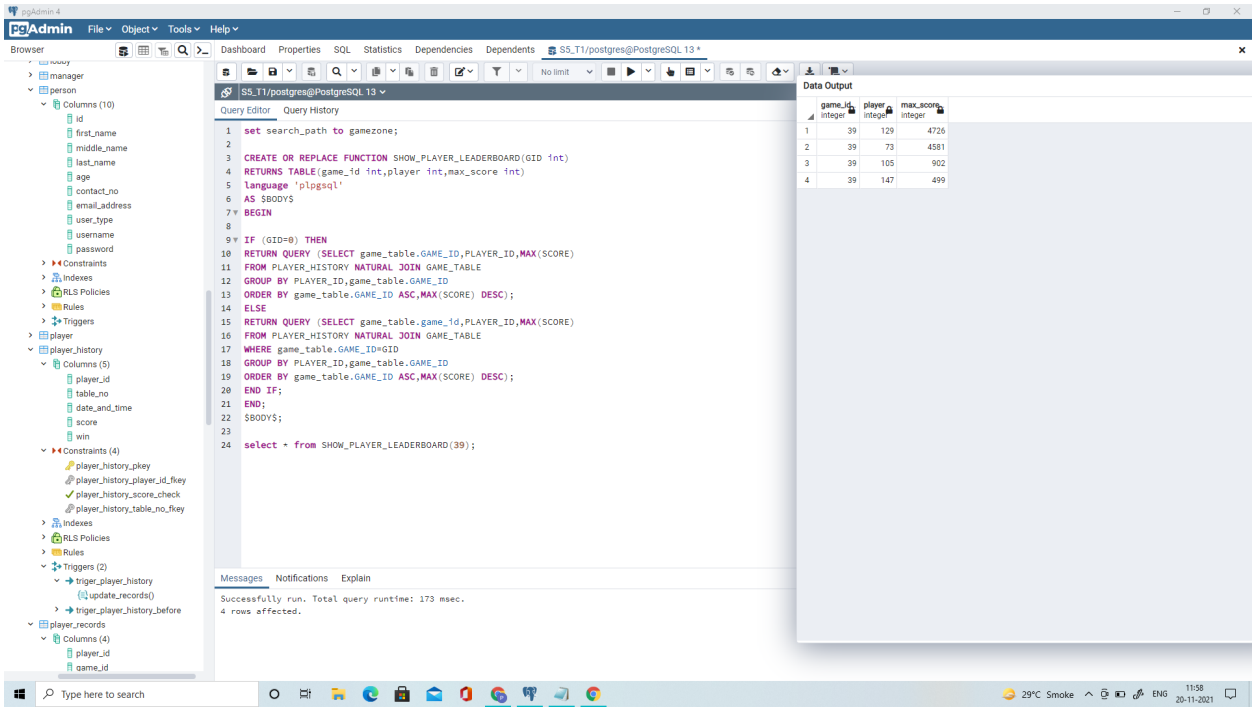


id	first_name	type	total_games
2	Mariana	Racing	76
3	Alejandrin	Casino	93
7	Keven	Racing	100
9	Jacinto	Board	97
10	Delaney	Casino	91
11	Dominic	Sports	93
12	Molly	Adventure	96
13	Anabel	Casino	79
14	Coy	Board	97
15	Martina	Adventure	97
16	Kimberly	Action	99
17	Camden	Adventure	74

No. of tuple : 114

39. Show Leaderboard of game_id 39.

select * from SHOW_PLAYER_LEADERBOARD(39);




```
set search_path to gamezone;

CREATE OR REPLACE FUNCTION SHOW_PLAYER_LEADERBOARD(GID int)
RETURNS TABLE(game_id int,player int,max_score int)
language 'plpgsql'
AS $BODY$
BEGIN

IF (GID=0) THEN
RETURN QUERY (SELECT game_table.GAME_ID,PLAYER_ID,MAX(SCORE)
FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
GROUP BY PLAYER_ID,game_table.GAME_ID
ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
ELSE
RETURN QUERY (SELECT game_table.game_id,PLAYER_ID,MAX(SCORE)
FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
WHERE game_table.GAME_ID=GID
GROUP BY PLAYER_ID,game_table.GAME_ID
ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
END IF;
END;
$BODY$;

select * from SHOW_PLAYER_LEADERBOARD(39);
```

game_id integer	player integer	max_score integer
1	39	129	4726
2	39	73	4581
3	39	105	902
4	39	147	499

No. of tuple = 4;

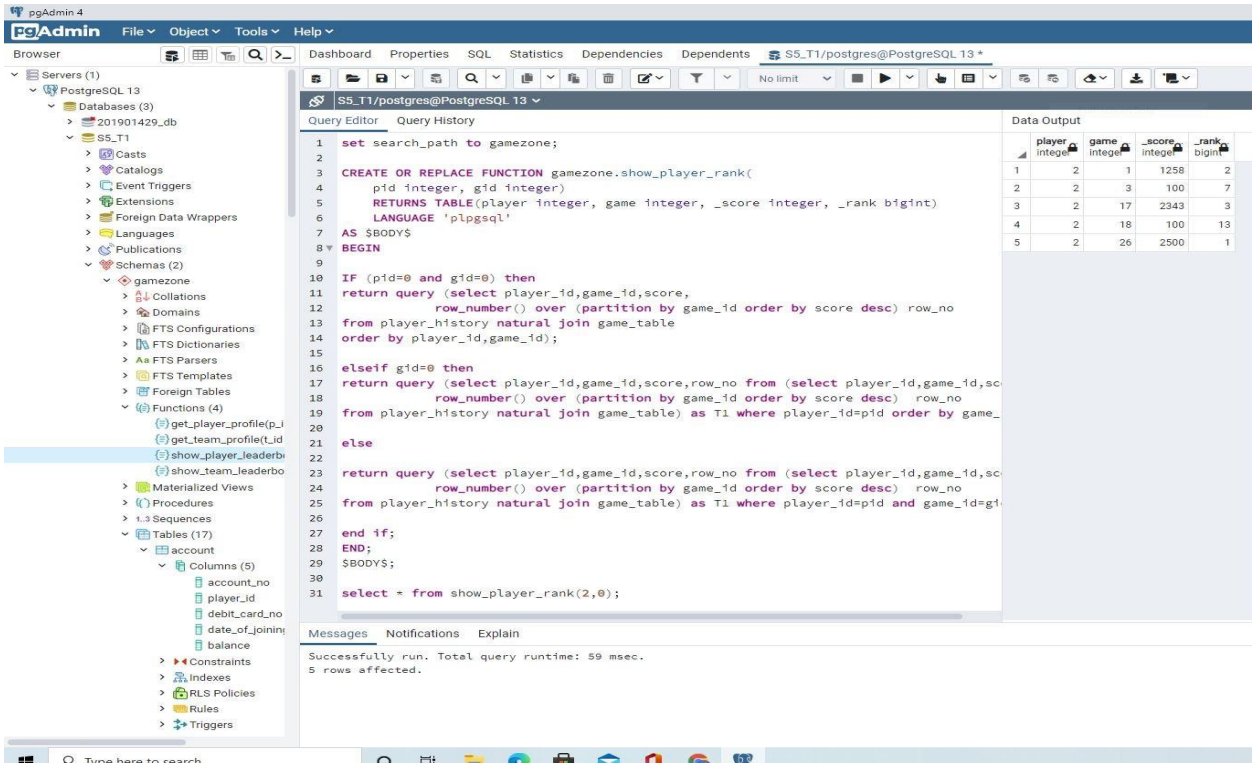
Function show_player_rank(pid integer, gid integer)

```
CREATE OR REPLACE FUNCTION gamezone.show_player_rank(
        pid integer, gid integer)
    RETURNS TABLE(player integer, game integer, _score integer, _rank
bigint)
    LANGUAGE 'plpgsql'
AS $BODY$
BEGIN

IF (pid=0 and gid=0) then
return query (select player_id,game_id,score,
                        row_number() over (partition by game_id order by
score desc) row_no
from player_history natural join game_table
order by player_id,game_id);

elseif gid=0 then
return query (select player_id,game_id,score,row_no from (select
player_id,game_id,score,
                        row_number() over (partition by game_id order by
score desc)  row_no
from player_history natural join game_table) as T1 where player_id=pid
order by game_id);

else

return query (select player_id,game_id,score,row_no from (select
player_id,game_id,score,
```

```
                    row_number() over (partition by game_id order by
score desc)  row_no
from player_history natural join game_table) as T1 where player_id=pid
and game_id=gid);


end if;
END;
$BODY$;
```

40. Show the rank of player_id 2 in each purchased game.

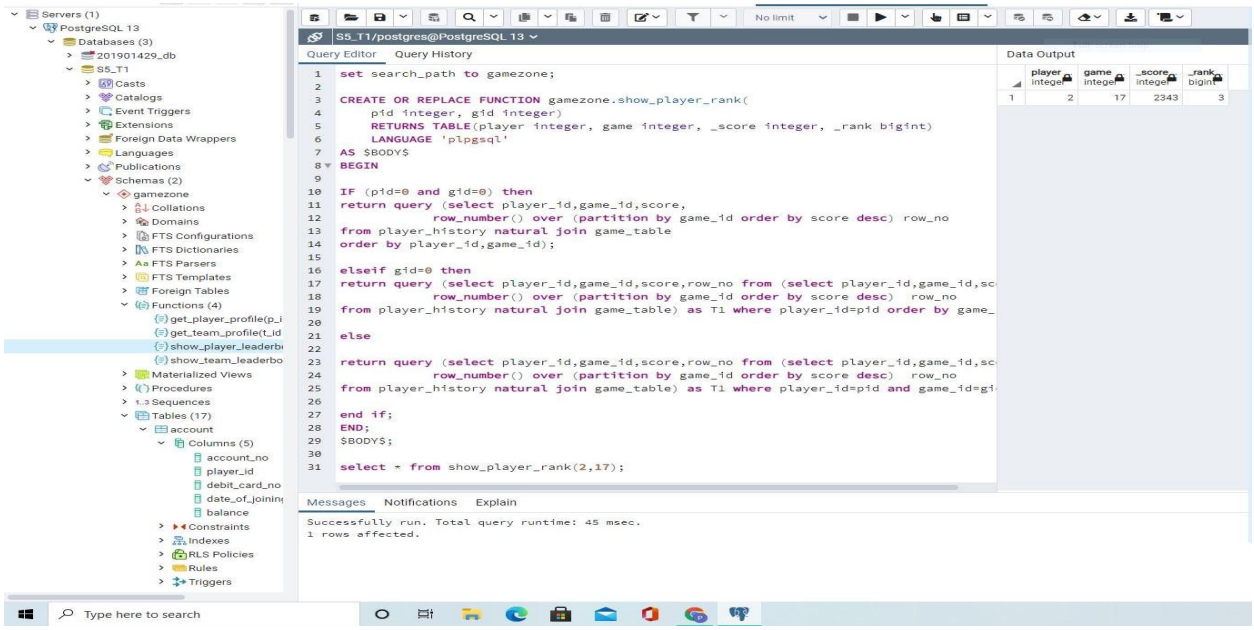
```
select * from show_player_rank(2,0);
```



No. of tuple: 5

41. Show the rank with the score of player 2 in game 17.

select * from show_player_rank(2,17);



No of tuple = 1

TRIGGERS:

1)

```
CREATE TRIGGER triger_player_history_to_transactions
    AFTER INSERT
    ON gamezone.player_history
    FOR EACH ROW
    EXECUTE FUNCTION gamezone.add_transactions_winloss();
```

2)

```
CREATE TRIGGER trigger_team_history_to_transactions
    AFTER INSERT
    ON gamezone.team_history
    FOR EACH ROW
    EXECUTE FUNCTION gamezone.add_transactions_winloss_team();
```

3)

```
CREATE TRIGGER triger_puchase
    BEFORE INSERT OR UPDATE
    ON gamezone.purchase
    FOR EACH ROW
    EXECUTE FUNCTION gamezone.check_game_requierements();
```

4)

```
CREATE TRIGGER triger_player_history_before
    BEFORE INSERT OR UPDATE
    ON gamezone.player_history
    FOR EACH ROW
    EXECUTE FUNCTION gamezone.check_purchased();
```

5)

```
CREATE TRIGGER triger_player_history
    AFTER INSERT
    ON gamezone.player_history
    FOR EACH ROW
    EXECUTE FUNCTION gamezone.update_records();
```

6)

```
CREATE TRIGGER trigger_updating
    AFTER INSERT
    ON gamezone.purchase
    FOR EACH ROW
    EXECUTE FUNCTION gamezone.update_transactions_pay();
```

TRIGGER FUNCTIONS:

1. add_transactions_winloss():-

```
CREATE FUNCTION gamezone.add_transactions_winloss()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
declare account_no char(11);
declare tr_id char(15);
declare dt timestamp;
declare ammount int;
BEGIN
account_no:= (select account.account_no from account natural join
player where player_id=new.player_id);
tr_id:= (SELECT array_to_string(array(select
substr('abcdefghijklmnopqrstuvxyzw0123456789',((random()*(36-1)+1)::int
eger),1) from generate_series(1,15)),''));

while tr_id in (select transaction_id from transactions) loop
tr_id:= (SELECT array_to_string(array(select
substr('abcdefghijklmnopqrstuvxyzw0123456789',((random()*(36-1)+1)::int
eger),1) from generate_series(1,15)),''));
end loop;

dt:= to_timestamp(to_char(now(),'YYYY-MM-DD HH:MI:SS'),'YYYY-MM-DD
HH:MI:SS');
```

```
ammount:= (select reward from game where game_id=(select game_id
from game_table where table_no=new.table_no));

if new.win=true then
insert into transactions values (tr_id, account_no, dt,ammount, 'win',
'Paytm');
update account
set balance=balance+ammount
where player_id=new.player_id;

if ammount<=(select balance from gamezone_balance) then
update gamezone_balance
set balance=balance-ammount;
else
raise notice 'Transaction should be added manually after enough balance
in gamezone_balance';
end if;

else
insert into transactions values (tr_id, account_no, dt,ammount/2, 'loss',
'Paytm');
update gamezone_balance
set balance=balance+ammount/2;
update account
set balance=balance-(ammount/2)
where player_id=new.player_id;
end if;
return NULL;
END;
$BODY$;
```

2. add_transactions_winloss_team():-

```
CREATE FUNCTION gamezone.add_transactions_winloss_team()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
declare account_no char(11);
declare tr_id char(15);
declare dt timestamp;
declare ammount int;
declare n int;
declare rec record;
BEGIN

dt:= to_timestamp(to_char(now(),'YYYY-MM-DD HH:MI:SS'),'YYYY-MM-DD
HH:MI:SS');
ammount:= (select reward from game where game_id=(select game_id
from game_table where table_no=new.table_no));
n:=(select no_players from team where team_id=new.team_id);

for rec in (select player_id from player where team_id=new.team_id)
loop
        account_no:= (select account.account_no from account natural
join player where player_id=rec.player_id);
        tr_id:= (SELECT array_to_string(array(select
substr('abcdefghijklmnopqrstuvxyzw0123456789',((random()*(36-1)+1)::int
eger),1) from generate_series(1,15)),''));
```

```
        while tr_id in (select transaction_id from transactions) loop
        tr_id:= (SELECT array_to_string(array(select
substr('abcdefghijklmnopqrstuvxyzw0123456789',((random()*(36-1)+1)::int
eger),1) from generate_series(1,15)),''));
        end loop;

        if new.win=true then
        insert into transactions values (tr_id, account_no, dt,ammount/n,
'win', 'Paytm');
        update account
        set balance=balance+(ammount/n)
        where player_id=rec.player_id;
        if ammount<=(select balance from gamezone_balance) then
        update gamezone_balance
        set balance=balance-ammount/n;
        else
        raise notice 'Transaction should be added manually after enough
balance in gamezone_balance';
        end if;
        else
        insert into transactions values (tr_id, account_no,
dt,ammount/(2*n), 'loss', 'Paytm');
        update gamezone_balance
        set balance=balance+ammount/(2*n);
        update account
        set balance=balance-(ammount/(2*n))
        where player_id=rec.player_id;
        end if;
end loop;
```

```
return NULL;
END;
$BODY$;
```

3. check_game_requierements():-

```
CREATE FUNCTION gamezone.check_game_requierements()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
IF EXISTS(select player_id
from game,(select * from player join person on player_id=id) as T1
where player_id=new.player_id and game_id=new.game_id and
T1.age>game.age and T1.height>game.height and T1.weight>game.weight
and price<=(select balance from account where
player_id=new.player_id)) THEN
RETURN NEW;
ELSE
RAISE NOTICE 'YOU CAN NOT BUY THIS GAME AS REQUIREMNETS OF
GAME IS NOT SATISFIED. PLEASE TRY ANOTHER GAME.';
RETURN NULL;
END IF;
END;
$BODY$;
```

4. check_purchased():-

```
CREATE FUNCTION gamezone.check_purchased()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
IF EXISTS(select player_id,table_no from purchase natural join
game_table where player_id=new.player_id and table_no=new.table_no)
then RETURN NEW;
ELSE
RAISE NOTICE 'Player Has Not Buy This Game. Tell Player to Buy first this
Game';
RETURN NULL;
END IF;
END;
$BODY$;
```

5. update_records():-

```
CREATE FUNCTION gamezone.update_records()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
declare g_id int;
declare iswin int;
```

```
BEGIN
g_id := (SELECT gid FROM (select game.game_id as gid,table_no from
game join game_table on game.game_id=game_table.game_id) as T1
where table_no=new.table_no);
if new.win=false then iswin:=0; else iswin:=1; end if;
IF EXISTS(SELECT player_id,game_id FROM player_records WHERE
player_id=new.player_id and game_id=g_id)
then
update player_records
set total_games=total_games+1, wins=wins+iswin
where player_id=new.player_id and game_id=g_id;
raise notice 'Value: %',g_id;
return NULL;
else
insert into player_records values(new.player_id,g_id,1,iswin);
return NULL;
end if;
return NULL;
END;
$BODY$;
```

6. update_transactions_pay():-

```
CREATE FUNCTION gamezone.update_transactions_pay()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
declare account_no char(11);
```

```
declare tr_id char(15);
declare dt timestamp;
declare ammount int;
BEGIN
account_no:= (select account.account_no from account natural join
player where player_id=new.player_id);
tr_id:= (SELECT array_to_string(array(select
substr('abcdefghijklmnopqrstuvxyzw0123456789',((random()*(36-1)+1)::int
eger),1) from generate_series(1,15)),''));

while tr_id in (select transaction_id from transactions) loop
tr_id:= (SELECT array_to_string(array(select
substr('abcdefghijklmnopqrstuvxyzw0123456789',((random()*(36-1)+1)::int
eger),1) from generate_series(1,15)),''));
end loop;
dt:= to_timestamp(to_char(now(),'YYYY-MM-DD HH:MI:SS'),'YYYY-MM-DD
HH:MI:SS');
ammount:= (select price from game where game_id=new.game_id);
insert into transactions values (tr_id, account_no, dt,ammount, 'pay',
'Cash');
update gamezone_balance
set balance=balance+ammount;
return NULL;
END;
$BODY$;
```

FUNCTIONS:

1)

```
CREATE OR REPLACE FUNCTION gamezone.get_player_profile(
    p_id integer)
RETURNS TABLE(player_id integer, first_name character varying,
middle_name character varying, last_name character varying, age integer,
height integer, weight integer, total_games_played integer, total_games_won
integer, winrate integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
create temp table profile(player_id int,first_name varchar(20),middle_name
varchar(20),last_name varchar(20),age int,height int,weight
int,total_games_played int,total_games_won int,winrate int) on commit drop;
insert into
profile(player_id,first_name,middle_name,last_name,age,height,weight) select
player.player_id,person.first_name,person.middle_name,person.last_name,perso
n.age,player.height,player.weight from person join player on
person.id=player.player_id where player.player_id=p_id;
--insert into profile(total_games_played,total_games_won,winrate) select
sum(total_games),sum(wins), (sum(total_games)*100)/sum(wins) from
player_records where player_records.player_id=p_id group by
player_records.player_id;
update profile
```

```
set (total_games_played,total_games_won,winrate)=(select
sum(total_games),sum(wins), (sum(wins)*100)/sum(total_games) from
player_records where player_records.player_id=p_id group by
player_records.player_id)
where profile.player_id=p_id;
return query table profile;
END;
$BODY$;
```

2)

```
   CREATE OR REPLACE FUNCTION gamezone.get_team_profile(
       t_id integer)
    RETURNS TABLE(team_id integer, team_name character varying,
no_of_players integer, total_games_played integer, total_games_won integer,
winrate integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
create temp table profile(team_id int,team_name varchar(20),no_of_players
int,total_games_played int,total_games_won int,winrate int) on commit drop;
insert into profile(team_id,team_name,no_of_players) select
team.team_id,team.t_name,team.no_players from team where
team.team_id=t_id;
update profile
```

```
set (total_games_played,total_games_won,winrate)=(select
sum(total_games),sum(wins), (sum(wins)*100)/sum(total_games) from
team_records where team_records.team_id=t_id group by
team_records.team_id)
where profile.team_id=t_id;
return query table profile;
END;
$BODY$;
```

3)

```
CREATE OR REPLACE FUNCTION gamezone.show_player_leaderboard(
    gid integer)
RETURNS TABLE(game_id integer, player integer, max_score integer)
LANGUAGE 'plpgsql'
COST 100
VOLATILE PARALLEL UNSAFE
ROWS 1000

AS $BODY$
BEGIN

IF (GID=0) THEN
RETURN QUERY (SELECT game_table.GAME_ID,PLAYER_ID,MAX(SCORE)
FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
GROUP BY PLAYER_ID,game_table.GAME_ID
ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
ELSE
RETURN QUERY (SELECT game_table.game_id,PLAYER_ID,MAX(SCORE)
FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
WHERE game_table.GAME_ID=GID
```

```
GROUP BY PLAYER_ID,game_table.GAME_ID
ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
END IF;
END;
$BODY$;
```

4)

```
CREATE OR REPLACE FUNCTION gamezone.show_team_leaderboard(
    gid integer)
RETURNS TABLE(team integer, max_score integer)
LANGUAGE 'plpgsql'
COST 100
VOLATILE PARALLEL UNSAFE
ROWS 1000

AS $BODY$
BEGIN

IF (GID=0) THEN
RETURN QUERY (SELECT GAME_ID,team_id,MAX(SCORE)
FROM TEAM_HISTORY NATURAL JOIN GAME_TABLE
GROUP BY TEAM_ID,GAME_ID
ORDER BY GAME_ID ASC,MAX(SCORE) DESC);
ELSE
RETURN QUERY (SELECT TEAM_ID,MAX(SCORE)
FROM TEAM_HISTORY NATURAL JOIN GAME_TABLE
WHERE GAME_ID=GID
GROUP BY TEAM_ID,GAME_ID
ORDER BY GAME_ID ASC,MAX(SCORE) DESC);
END IF;
```

```
END;
$BODY$;
```

5)

```
CREATE OR REPLACE FUNCTION gamezone.show_player_rank(
    pid integer,
    gid integer)
RETURNS TABLE(player integer, game integer, _score integer, _rank bigint)
LANGUAGE 'plpgsql'
COST 100
VOLATILE PARALLEL UNSAFE
ROWS 1000

AS $BODY$
BEGIN

IF (pid=0 and gid=0) then
return query (select player_id,game_id,score,
                    row_number() over (partition by game_id order by score
desc) row_no
from player_history natural join game_table
order by player_id,game_id);

elseif gid=0 then
return query (select player_id,game_id,score,row_no from (select
player_id,game_id,score,
                    row_number() over (partition by game_id order by score
desc)  row_no
from player_history natural join game_table) as T1 where player_id=pid order by
game_id);
```

```
else

return query (select player_id,game_id,score,row_no from (select
player_id,game_id,score,
                        row_number() over (partition by game_id order by score
desc)  row_no
from player_history natural join game_table) as T1 where player_id=pid and
game_id=gid);

end if;
END;
$BODY$;
```

6)

```
   CREATE OR REPLACE FUNCTION gamezone.show_team_rank(
        tid integer,
        gid integer)
    RETURNS TABLE(player integer, game integer, _score integer, _rank bigint)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN

IF (tid=0 and gid=0) then
return query (select team_id,game_id,score,
```

```
                    row_number() over (partition by game_id order by score
desc) row_no
from team_history natural join game_table
order by team_id,game_id);

elseif gid=0 then
return query (select team_id,game_id,score,row_no from (select
team_id,game_id,score,
                    row_number() over (partition by game_id order by score
desc)  row_no
from team_history natural join game_table) as T1 where team_id=tid order by
game_id);

else

return query (select team_id,game_id,score,row_no from (select
team_id,game_id,score,
                    row_number() over (partition by game_id order by score
desc)  row_no
from team_history natural join game_table) as T1 where team_id=tid and
game_id=gid);

end if;
END;
$BODY$;
```