

# 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';
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under 'Tables (17)', including 'account', 'game', 'game\_table', 'gamezone\_balance', 'lobby', 'manager', 'person', 'player', 'player\_history', 'player\_records', 'purchase', 'roles', 'team', 'team\_history', 'team\_records', 'transactions', 'worker'. The main area shows a query editor with the following SQL code:

```
1 set search_path to gamezone;  
2  
3 select player_id from account  
4 where date_of_joining >= '1/1/2019';
```

The results pane shows a table with one column 'player\_id' and 10 rows of data:

player_id
1
2
3
4
5
6
7
8
9
10

No of tuples = 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;
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar indicates the connection is to S5\_T1/postgres@PostgreSQL 13 \*. The main area displays the following SQL code:

```
1 set search_path to gamezone;  
2  
3 select user_type, count(*)  
4 from person  
5 group by user_type;
```

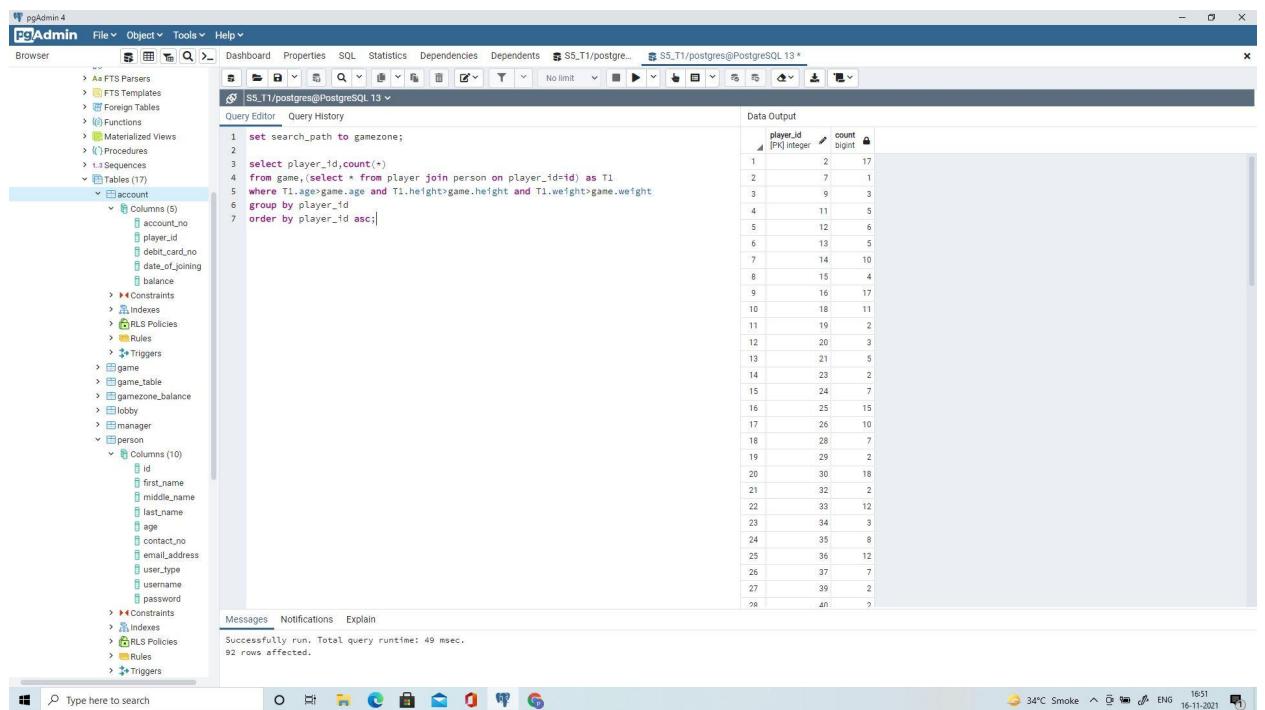
Below the code, the results are shown in a table:

	user_type	count
1	worker	31
2	manager	5
3	player	114

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;
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The browser pane on the left lists database objects like FTS Parsers, FTS Templates, Functions, Procedures, Sequences, and Tables (17). The 'account' table is selected. The query editor contains the SQL code provided above. The data output pane shows a table with two columns: player\_id (PK Integer) and count (bigint). The table has 92 rows, corresponding to the 27 players listed in the 'person' table. The last row shows a total count of 92 for player\_id 27.

player_id	count
1	2
2	1
3	3
4	5
5	6
6	5
7	10
8	4
9	17
10	11
11	2
12	3
13	5
14	2
15	7
16	15
17	10
18	7
19	2
20	18
21	2
22	12
23	3
24	8
25	12
26	7
27	2
28	9

Messages Notifications Explain  
Successfully run. Total query runtime: 49 msec.  
92 rows affected.

No of tuple = 92

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

Select \* from person

Where age>=18 and user\_type = 'player';

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like lobby, manager, person, and player. The Query Editor window contains the following SQL code:

```
1 set search_path to "S5_T1";
2
3 select * from person
4 where age>=18 and user_type='player';
5
```

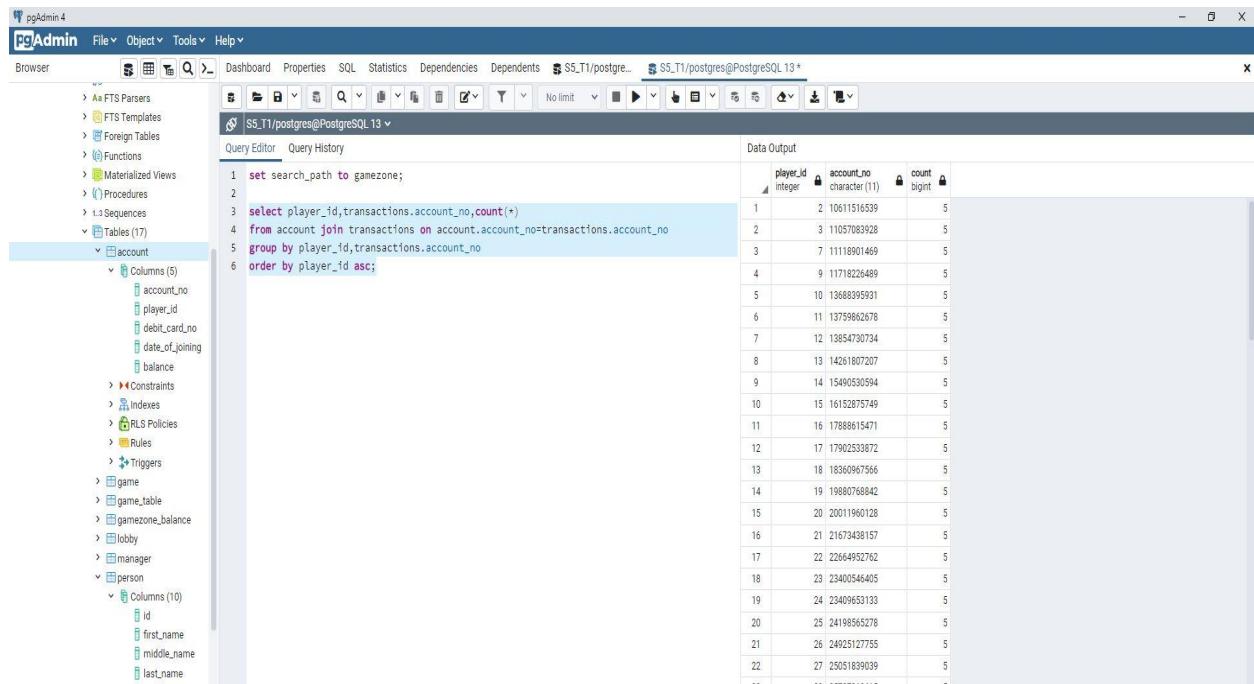
The results pane shows a table with 93 rows of data, corresponding to the query results. A green message bar at the bottom right indicates the query was successfully run and completed in 68 msec.

	id	first_name	middle_name	last_name	age	contact_no	email_address	user_type	username	password
87	135	Lew	omnis	Kuhn	51	1310560325	powlowski.winona@example.com	player	rosina17	
88	136	Nyasia	corrupti	Ratke	28	2609096083	crooks.bryan@example.com	player	maryjane76	
89	137	Brent	ea	Koepf	64	9140532490	wbreiterberg@example.com	player	katherine87	
90	142	Mohammed	et	Hickle	24	5445732383	abigail22@example.com	player	dietrich.belle	
91	145	Tony	dolores	Hamill	38	4288727961	zulauf.style@example.com	player	jakubowski.orie	
92	147	Annabel	qui	Dickens	18	733159618	maureen98@example.com	player	nitzsche.vicente	
93	149	Tate	et	Murray	34	9659913187	funk.lowell@example.com	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;
```



The screenshot shows the pgAdmin 4 interface with the following details:

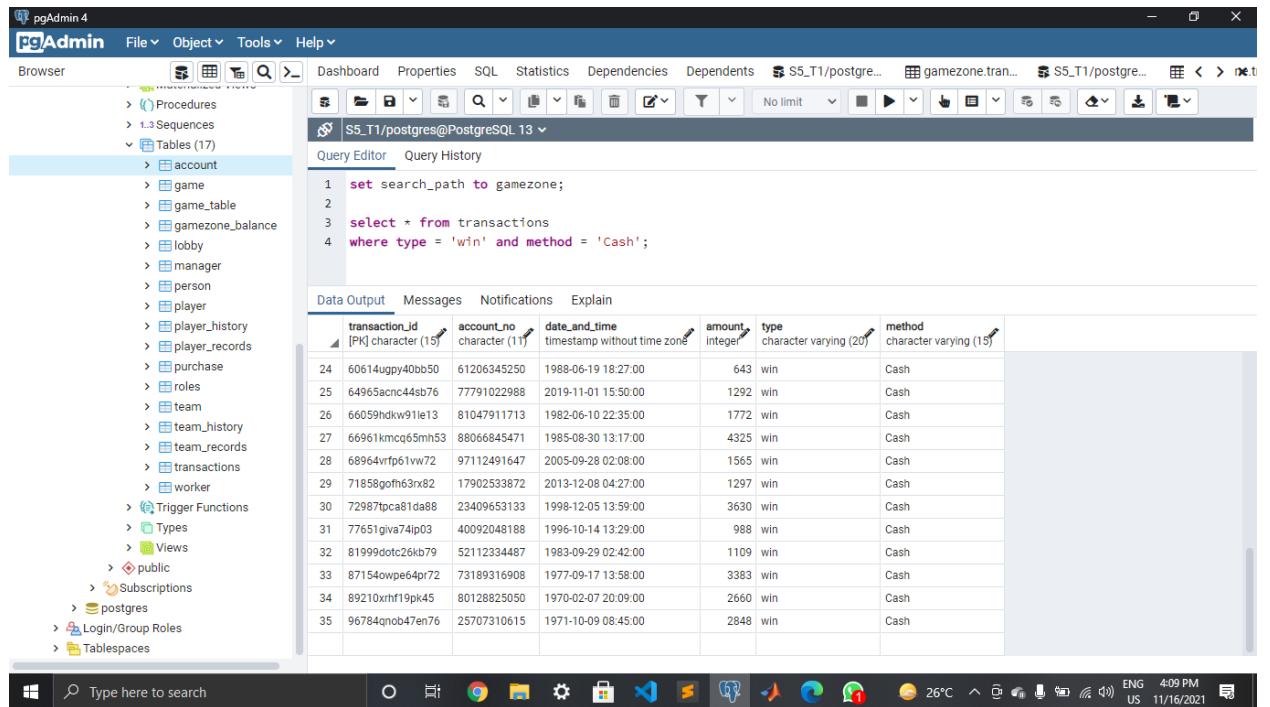
- Browser:** Shows the database schema structure, including tables like game, game\_table, lobby, manager, person, and account.
- Query Editor:** Displays the SQL query entered by the user.
- Data Output:** Shows the results of the query, which consists of 114 tuples. The columns are player\_id, account\_no, and count.

player_id	account_no	count
1	10611516539	5
2	3 11057083928	5
3	7 1111891469	5
4	9 1171826489	5
5	10 136888395931	5
6	11 13759862678	5
7	12 13854730734	5
8	13 14261607207	5
9	14 15490530594	5
10	15 16152875749	5
11	16 17888615471	5
12	17 17902333872	5
13	18 18360967556	5
14	19 19880768842	5
15	20 20011960128	5
16	21 21673438157	5
17	22 22664952762	5
18	23 23400546405	5
19	24 23409653133	5
20	25 24198565278	5
21	26 24925127755	5
22	27 25051839039	5
...	...	...

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';
```



The screenshot shows the pgAdmin 4 interface with a query editor window displaying the following SQL code:

```
1 set search_path to gamezone;  
2  
3 select * from transactions  
4 where type = 'win' and method = 'Cash';
```

The results are shown in a Data Output table:

transaction_id	account_no	date_and_time	amount	type	method
24	60614ugpy40bb50	1988-06-19 18:27:00	643	win	Cash
25	64965acnc44sb76	2019-11-01 15:50:00	1292	win	Cash
26	66059hdkw1le13	1982-06-10 22:35:00	1772	win	Cash
27	66961kmcq65mh53	1985-08-30 13:17:00	4325	win	Cash
28	68964vrp61w72	2005-09-28 02:08:00	1565	win	Cash
29	71858gofhb3rx82	2013-12-08 04:27:00	1297	win	Cash
30	72987tpca81da88	1998-12-05 13:59:00	3630	win	Cash
31	77651giw74lp03	1996-10-14 13:29:00	988	win	Cash
32	81999dotc26kb79	1983-09-29 02:42:00	1109	win	Cash
33	87154owpe64pr72	1977-09-17 13:58:00	3383	win	Cash
34	89210xrhf19pk45	1970-02-07 20:09:00	2660	win	Cash
35	96784qnob47en76	1971-10-09 08:45:00	2848	win	Cash

No of tuple = 35

## 7. Count the number of workers for each role.

```
SELECT role, COUNT(*)
```

```
FROM worker
```

```
GROUP BY role;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects: Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (17), Roles, Team, Team History, Team Records, Transactions, and Worker. Under 'Worker', it shows Columns (2) with 'worker\_id' and 'role'. The main area is a 'Query Editor' window with the following content:

```
role          count
character varying(20)  bigint
1  voluptate    4
2  ab           3
3  porro         5
4  est           5
5  quia          4
6  laboriosam   2
7  et             1
8  nobis         4
9  ipsa           2
10 molestiae    1
```

The 'Messages' panel at the bottom right indicates: "Successfully run. Total query runtime: 401 msec. 10 rows affected."

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
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like 'player' and 'player\_history'. The main area shows the query editor with the following SQL code:

```
1 set search_path to "S5_T1";
2
3 select * from (person join player on person.id = player.player_id)
4 where player.team_id=7;
```

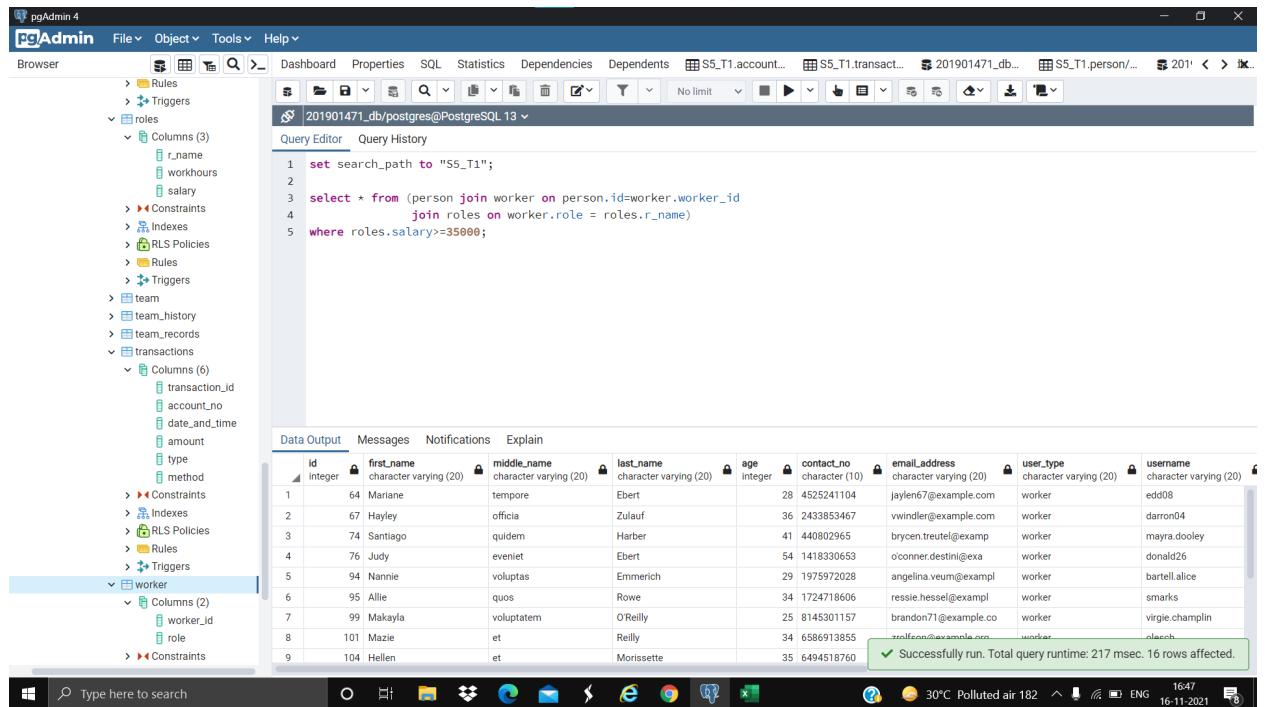
The results pane shows a table with 4 rows of data:

	<u><a href="#">id</a></u>	<u><a href="#">first_name</a></u>	<u><a href="#">middle_name</a></u>	<u><a href="#">last_name</a></u>	<u><a href="#">age</a></u>	<u><a href="#">contact_no</a></u>	<u><a href="#">email_address</a></u>	<u><a href="#">user_type</a></u>	<u><a href="#">username</a></u>
1	25	Dax	necessitatibus	Crooks	42	6624748290	joel.koch@example.or	player	kovacek.jordon
2	57	Cleta	reprehenderit	Legros	15	1958453160	gmoscisk@example.co	player	lia.hermann
3	60	Cristian	qui	Wisozk	70	1245369794	lowell.spinka@examplepl	player	theresia50
4	73	Corrine	autem	Runolfsdottir	60	802333377	cmurazik@example.net	player	delphine.gulgowski

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;
```



The screenshot shows the pgAdmin 4 interface with a query executed in the Query Editor. The query selects data from three tables: person, worker, and roles, where the worker's role has a salary of 35000 or more. The results are displayed in a Data Output table with 16 rows affected.

id	first_name	middle_name	last_name	age	contact_no	email_address	user_type	username
1	64	Mariane	tempore	Ebert	28	4525241104	worker	ed0d08
2	67	Hayley	officia	Zulauf	36	2433853467	worker	darron04
3	74	Santiago	quidem	Harber	41	440802965	worker	maya.dooley
4	76	Judy	eveniet	Ebert	54	1418330653	worker	donald26
5	94	Nannie	voluptas	Emmerich	29	1975972028	worker	bartell.alice
6	95	Allie	quo	Rowe	34	1724718606	worker	smarks
7	99	Makayla	voluptatem	O'Reilly	25	8145301157	worker	virgie.champlin
8	101	Mazie	et	Reilly	34	6586913855	worker	zolaicon@example.org
9	104	Hellen	et	Morissette	35	6494518760	worker	olarch

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;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various tables and their columns. The central area is the Query Editor where the following SQL query is entered:

```
1 set search_path to gamezone;  
2  
3 with tem as (select account_no,count(*)  
4 from transactions  
5 group by account_no  
6 having count(*)>=5)  
7  
8 select first_name,middle_name,last_name,contact_no,email_address  
9 from person join (account join tem on account.account_no=tem.account_no) as T1 on pe
```

The Data Output pane shows the results of the query, which consists of 44 rows of data. The columns in the result are:

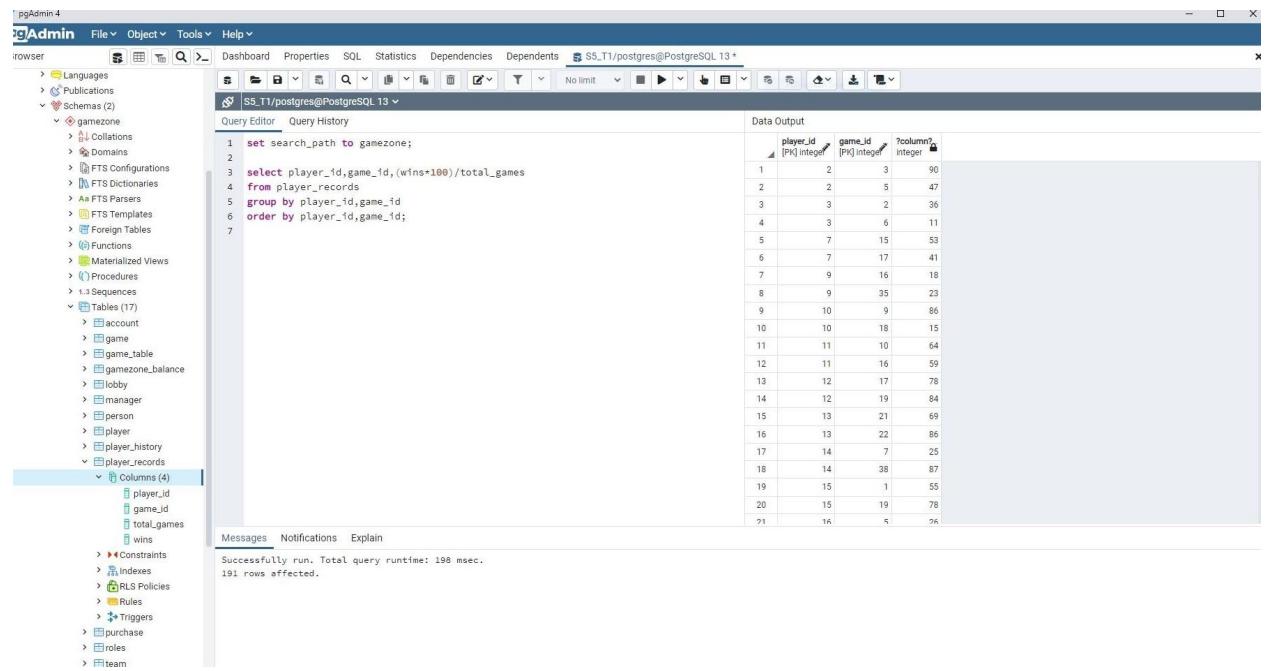
first_name	middle_name	last_name	contact_no	email_address
Uwein	et	Hermann	6185172785	linda13@example.org
Jairo	sed	Kohler	288224477	runt.zula@example.o
Enola	est	Leannon	315349017	lang.julio@example.o
Adolph	dignissimos	Oronin	6683241140	kris.guillermo@examp
Shany	similique	Bartell	6820651122	dhayes@example.com
Aubree	dicta	Schroeder	446029621	erain35@example.net
Jose	repellendus	Stracke	9976825568	emmet.murphy@example
Sanford	qua	Treutel	4831261703	cconnelly@example.ne
Claude	autem	Kunze	278835520	kiara71@example.com
Jasen	eaque	Bradtke	8148626347	wilma10@example.com
Jovany	facere	Collins	5542341348	emie.moen@example.o
Vela	earum	Yost	4883088122	sydnee.mora@example
Loy	commodi	Crist	4145417314	bennie90@example.net
Clemens	et	Barrows	6950553057	vulrich@example.org
Berry	nisi	Mraz	4081863219	ansley98@example.org
Leonel	provident	Jacobi	704126793	vbaumach@example.co
Rafael	nulla	Deckow	77735977	glockman@example.com
Sheila	corporis	Hettinger	4452534141	dschafer@example.co
Janae	esse	Marvin	2162934687	becker.maida@example
Waino	molestiae	Renner	7961881096	gregorio.pfammenti

Below the Data Output pane, the Messages, Notifications, and Explain tabs are visible. A message at the bottom indicates the query was successfully run and 44 rows were affected.

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;
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The left sidebar displays the database schema with various objects like Languages, Publications, Schemas, FTS Configurations, Functions, Procedures, Sequences, and Tables. The 'player\_records' table is selected, and its columns (player\_id, game\_id, total\_games, wins) are visible. The main area contains the SQL query:

```
1 set search_path to gamezone;  
2  
3 select player_id,game_id,(wins*100)/total_games  
4 from player_records  
5 group by player_id,game_id  
6 order by player_id,game_id;
```

The results are displayed in a Data Output table:

player_id	game_id	?column?
1	2	3
2	2	5
3	3	2
4	3	6
5	7	15
6	7	17
7	9	16
8	9	35
9	10	9
10	10	18
11	11	10
12	11	16
13	12	17
14	12	19
15	13	21
16	13	22
17	14	7
18	14	38
19	15	1
20	15	19
21	16	5

Below the table, the message bar indicates: "Successfully run. Total query runtime: 198 msec. 191 rows affected."

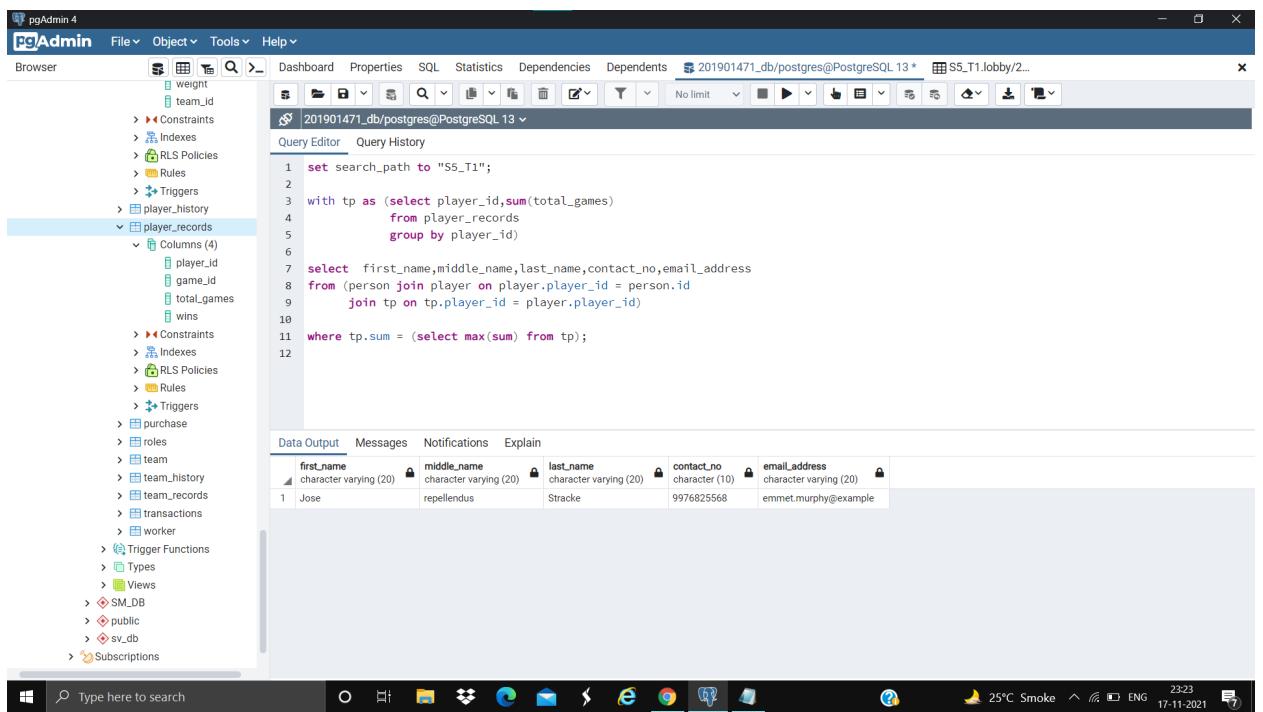
No. of tuples = 191.

## 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);
```



The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database schema structure, including tables like team, player\_records, and player.
- Query Editor:** Displays the SQL query:

```
1 set search_path to "S5_T1";
2
3 with tp as (select player_id,sum(total_games)
4                 from player_records
5                 group by player_id)
6
7 select first_name,middle_name,last_name,contact_no,email_address
8 from (person join player on player.player_id = person.id
9           join tp on tp.player_id = player.player_id)
10
11 where tp.sum = (select max(sum) from tp);
12
```

- Data Output:** Shows the result of the query, which is a single tuple:

	first_name	middle_name	last_name	contact_no	email_address
1	Jose	repellendus	Stracke	9976825568	emmet.murphy@example

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';
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with the 'worker' table selected. The main area contains a query editor with the following SQL code:

```
1 set search_path to "S5_T1";
2
3 select * from (person join worker on worker.worker_id=person.id)
4 where worker.role = 'ab';
5
```

Below the query editor is a data grid titled 'Data Output' showing the results of the query:

	<b>id</b>	<b>first_name</b>	<b>middle_name</b>	<b>last_name</b>	<b>age</b>	<b>contact_no</b>	<b>email_address</b>	<b>user_type</b>	<b>username</b>	<b>password</b>
1	98	Brannon	dolor	Miller	63	7184178264	ablock@example.net	worker	iweber	aac10
2	126	Lew	et	Kohler	64	3015665515	callista42@example.or	worker	emard.albertha	212eb
3	141	Moses	quia	Ruecker	32	6472282373	prohaska.twila@examp	worker	sdaniel	c6693

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);
```

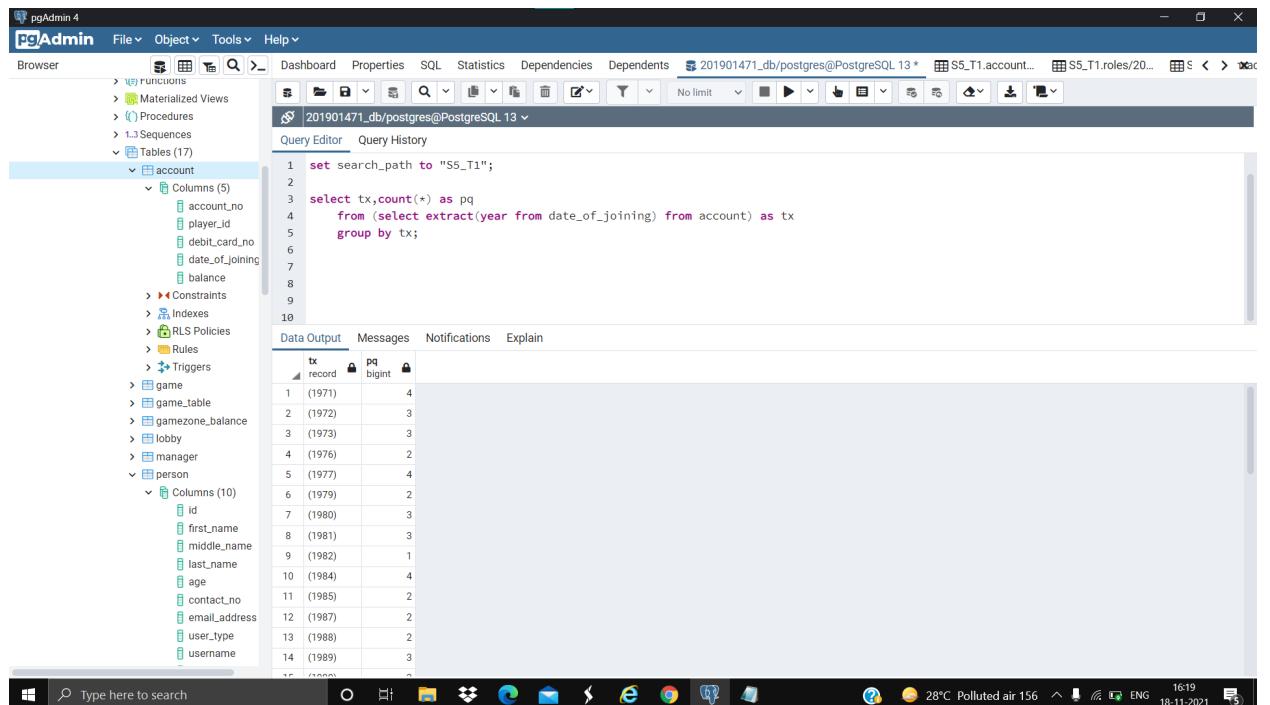
```
1 set search_path to "S5_T1";
2
3 with tp as (select team_id,sum(total_games)
4   from team_records
5   group by team_id)
6
7 select first_name,middle_name,last_name,contact_no,email_address
8   from (person join player on player.player_id = person.id
9         join tp on tp.team_id = player.team_id)
10
11 where tp.sum = (select max(sum) from tp);
12
```

first_name	middle_name	last_name	contact_no	email_address
Waino	molestiae	Renner	7961881096	gregorio.pfannerstil
Esmeralda	odio	Osinski	8795512587	kschumm@example.com
Destiney	quisquam	Robel	7372613941	fjacobsen@example.co

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;
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
set search_path to "S5_T1";  
select tx,count(*) as pq  
from (select extract(year from date_of_joining) from account) as tx  
group by tx;
```

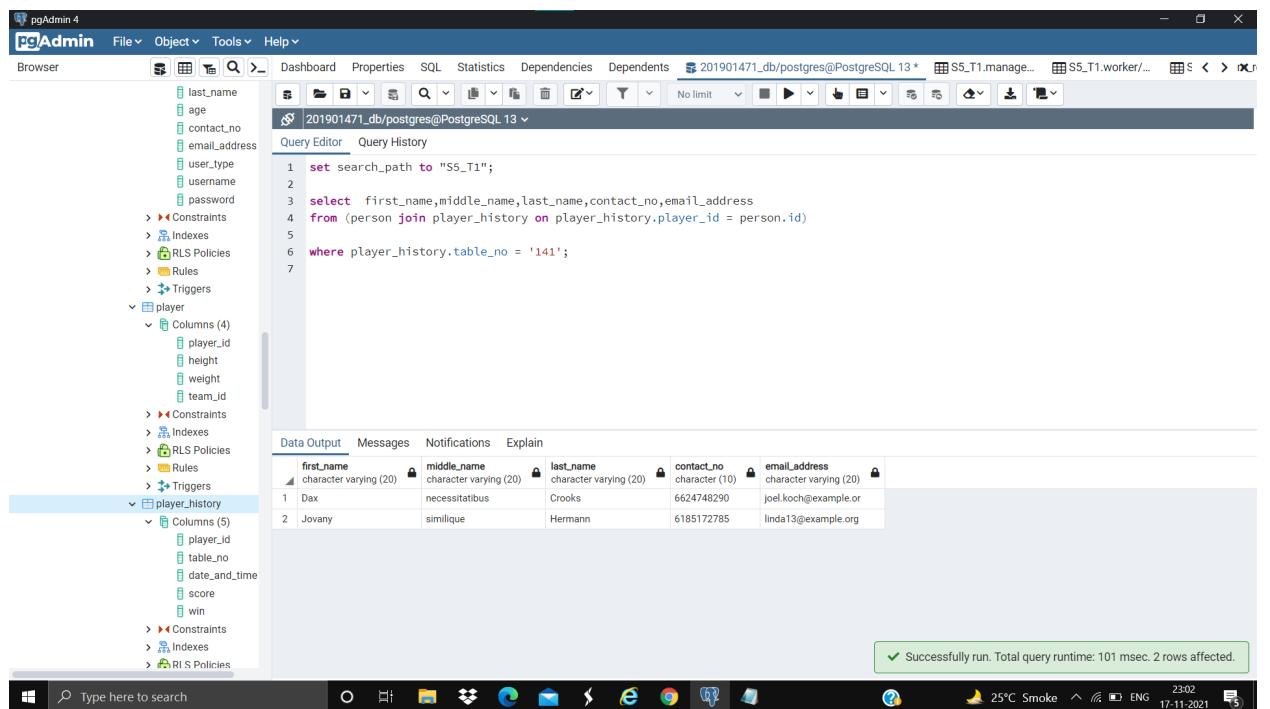
The results table shows the count of tuples for each year:

tx	pq
1 (1971)	4
2 (1972)	3
3 (1973)	3
4 (1976)	2
5 (1977)	4
6 (1979)	2
7 (1980)	3
8 (1981)	3
9 (1982)	1
10 (1984)	4
11 (1985)	2
12 (1987)	2
13 (1988)	2
14 (1989)	3
15 (1990)	0

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';
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
1 set search_path to "S5_T1";  
2  
3 select first_name,middle_name,last_name,contact_no,email_address  
4 from (person join player_history on player_history.player_id = person.id)  
5  
6 where player_history.table_no = '141';  
7
```

The results table shows two rows:

	first_name	middle_name	last_name	contact_no	email_address
1	Dax	necessitatibus	Crooks	6624748290	joel.koch@example.or
2	Jovany	similique	Hermann	6185172785	linda13@example.org

A message at the bottom right indicates: "Successfully run. Total query runtime: 101 msec. 2 rows affected."

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';

```

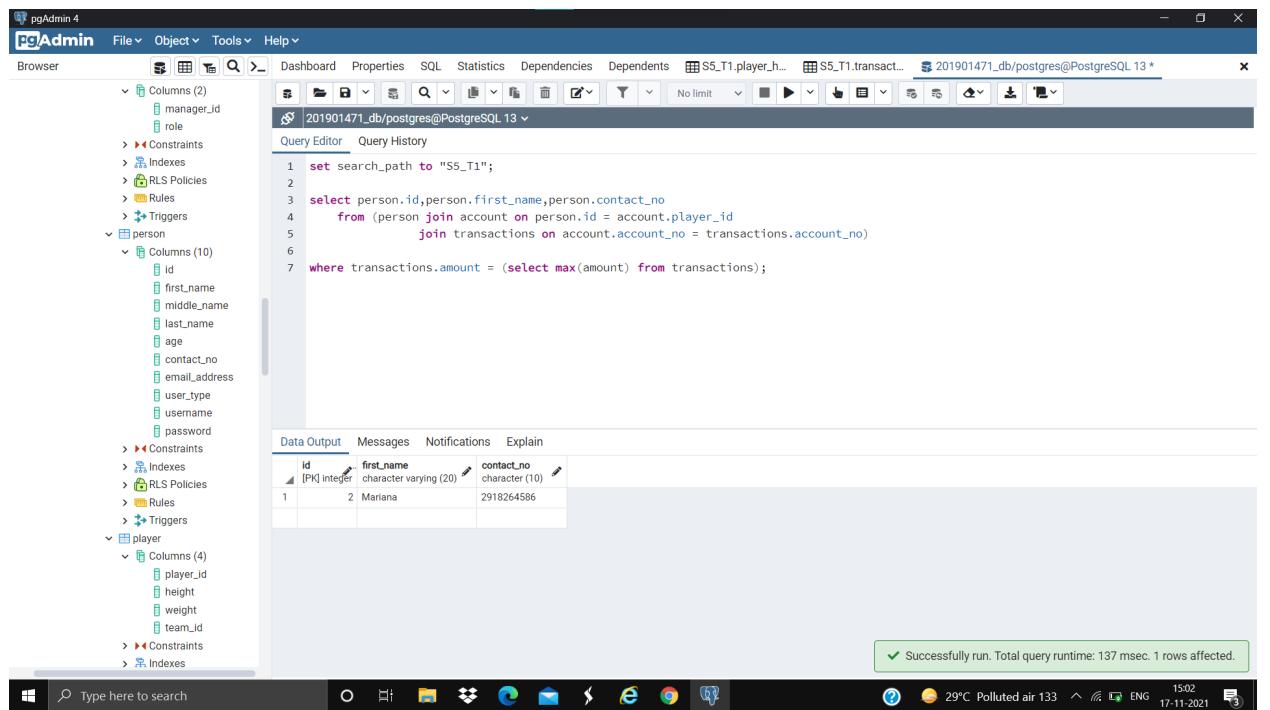
player_id	first_name	middle_name	last_name	game_name	contact_no	email_address
2	Jairo	sed	Kohler	accusantium	28822444//	kulas.dayana@example.o
3	Dominic	nisi	Ermer	saepe	4711281324	edith88@example.net
4	Alden	et	Kovacek	saepe	6208587407	hayden93@example.org
5	Andreane	mollitia	Murphy	saepe	2622139655	efrain35@example.net
6	Aubree	dicta	Schroeder	saepe	446029621	wbreitenberg@example
7	Brent	ea	Koepf	fuga	9140532490	asia.kilback@example
8	Alejandrin	deleniti	Daugherty	fuga	8683979785	hyundt.bogen@example
9	Brigid	doloremque	Beier	fuga	1328147048	clifton.bogan@example
10	Clifford	occaecati	Pfeffer	fuga	1099592486	waelchi.lord@example
11	Shvann	distinctio	Wilkinson	aut	411993458	

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);



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
set search_path to "S5_T1";  
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);
```

The results table shows one row:

id	first_name	contact_no
1	Mariana	2918264586

A green message bar at the bottom right indicates: "Successfully run. Total query runtime: 137 msec. 1 rows affected."

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;
```

The screenshot shows the pgAdmin 4 interface. On the left is the object browser tree, which includes nodes for Triggers, game\_table, Constraints, gamezone, RLS Policies, Rules, Triggers, gamezone\_balance, lobby, manager, person, player, and player\_history. The person node is currently selected, showing its columns: id, first\_name, middle\_name, last\_name, age, contact\_no, email\_address, user\_type, username, and password. The right side of the window has a 'Query Editor' tab containing the following SQL code:

```
1 set search_path to gamezone;  
2  
3 select player_id, count(*) game_purchased  
4 from purchase  
5 group by player_id  
6 order by player_id asc;
```

Below the code, the 'Messages' tab shows the output: "Successfully run. Total query runtime: 46 msec. 100 rows affected." To the right, the 'Data Output' pane displays the results of the query as a table:

player_id	game_purchased
81	100
82	102
83	103
84	105
85	106
86	107
87	109
88	110
89	112
90	113
91	114
92	115
93	116
94	117
95	118
96	119
97	120
98	121
99	123
100	128

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';
```

```
set search_path to "SS_T1";
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';
```

player_id	first_name	middle_name	last_name	game_name	contact_no	email_address
85	Opal	qui	Kirlin	sunt	243221775	jamaal.trantow@example.com
102	Leonor	nisi	Heidenreich	sunt	286655964	kfrenney@example.net
73	Corinne	autem	Runolfsdottir	sunt	802333377	cmurazik@example.net
116	Lloyd	quae	Durgan	sunt	361733370	corwin.aletha@example.org
100	Jessie	quis	Hauck	sunt	2050527472	mose25@example.org
37	Sanford	qua	Treutel	sunt	4831261703	cconnelly@example.net

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;
```

The screenshot shows the PGAdmin interface. On the left is a tree view of database objects under 'gamezone'. In the center is the 'Query Editor' tab containing the SQL query. To the right is the 'Data Output' tab showing the results of the query.

game_id	no_of_tables
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40

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';
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like game, game\_table, gamezone\_balance, lobby, manager, and person. The person table is currently selected, showing its 10 columns: id, first\_name, middle\_name, last\_name, age, contact\_no, email\_address, user\_type, username, and password. The right pane contains a Query Editor window with the following SQL code:

```
1 set search_path to "S5_T1";
2
3 select person.id, person.first_name, person.middle_name, person.last_name, person.contact_no, person.email_address, account.balance
4   from ( person join account on person.id = account.player_id )
5 where account.balance >= '50000';
6
```

Below the query editor, the Data Output tab is selected, showing the results of the query. The results are as follows:

	id	first_name	middle_name	last_name	contact_no	email_address	balance
1	3	Alejandrín	ut	Hoeger	7800501447	gilberto.lemke@example.com	73895
2	7	Keven	fuga	Grady	1817842697	blaise82@example.org	53184
3	11	Dominic	nisi	Ernsler	4711281324	kulas.dayana@example.com	67497
4	13	Anabel	consequatur	Rohan	5626135839	raegan84@example.com	88742
5	16	Kimberly	temporibus	Vandervort	9587970338	aromaguera@example.n	59658
6	18	Collin	qua	Conn	3480646000	dooley.christina@exa	61723
7	19	Carlos	est	Frami	3863436015	kirstin.bosco@example.com	50738
8	20	Joel	totam	Brekke	2645835183	cassin.ettie@example.com	56886
9	24	Rodrigo	qui	Witting	9135283110	ofell@example.com	77790

No. of tuple : 49

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

```
SELECT transactions.method, COUNT(*)
```

```
FROM transactions
```

```
GROUP BY transactions.method
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under the 'transactions' schema, including columns like transaction\_id, account\_no, date\_and\_time, amount, type, and method. The 'method' column is highlighted. In the center, the 'Query Editor' pane shows the following SQL query and its results:

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

method	count
Gpay	139
Paytm	135
Cash	122
Check	104

The 'Messages' pane at the bottom right indicates that the query was successfully run with a total runtime of 228 msec and 4 rows affected.

No of tuple = 4

**24. Find the number of games for each type.**

```
SELECT game.type, count(*)
```

```
FROM game GROUP BY game.type;
```

The screenshot shows the pgAdmin 4 interface. On the left is the 'Browser' pane, which displays the database schema for 'gamezone\_db'. It lists tables like player\_id, debit\_card\_no, date\_of\_joining, balance, and game. The 'game' table is selected, showing its columns: game\_id, game\_name, type, mode, price, reward, age, height, and weight. On the right is the 'Query Editor' pane, which contains the following SQL query:

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

The 'Data Output' tab is active, showing the results of the query:

type	count
Sports	9
Adventure	4
Racing	10
Casino	8
Action	5
Board	4

Below the results, the 'Messages' pane shows the output: "Successfully run. Total query runtime: 126 msec. 6 rows affected."

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)
```

The screenshot shows the pgAdmin 4 interface. On the left is the 'Browser' pane, which displays the database schema. Under the 'game' table, the 'Columns' section is expanded, showing columns: game\_id, game\_name, type, mode, price, reward, age, height, and weight. The 'reward' column is highlighted. The 'Messages' pane at the bottom right shows a successful query execution message: 'Successfully run. Total query runtime: 113 msec. 1 rows affected.'

game_id	game_name	type	mode	price	reward	age	height	weight
1	praesentium	Action	offline	57266	1000	39	191	44

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;
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects, including tables like account, game, game\_table, gamezone\_balance, lobby, manager, person, player, player\_history, player\_records, and purchase. The purchase table is expanded, showing its columns: purchase\_id, player\_id, game\_id, and no\_hours. The 'Query Editor' pane contains the following SQL code:

```
1 set search_path to gamezone;  
2  
3 select game_id, count(player_id) as no_of_players  
4 from purchase  
5 group by game_id;
```

The 'Data Output' pane shows the results of the query, which consists of 36 tuples. The table has two columns: game\_id (integer) and no\_of\_players (bigint). The data is as follows:

game_id	no_of_players
17	36
18	25
19	31
20	30
21	14
22	3
23	17
24	20
25	22
26	28
27	37
28	33
29	13
30	5
31	18
32	2
33	16
34	27
35	23
36	8

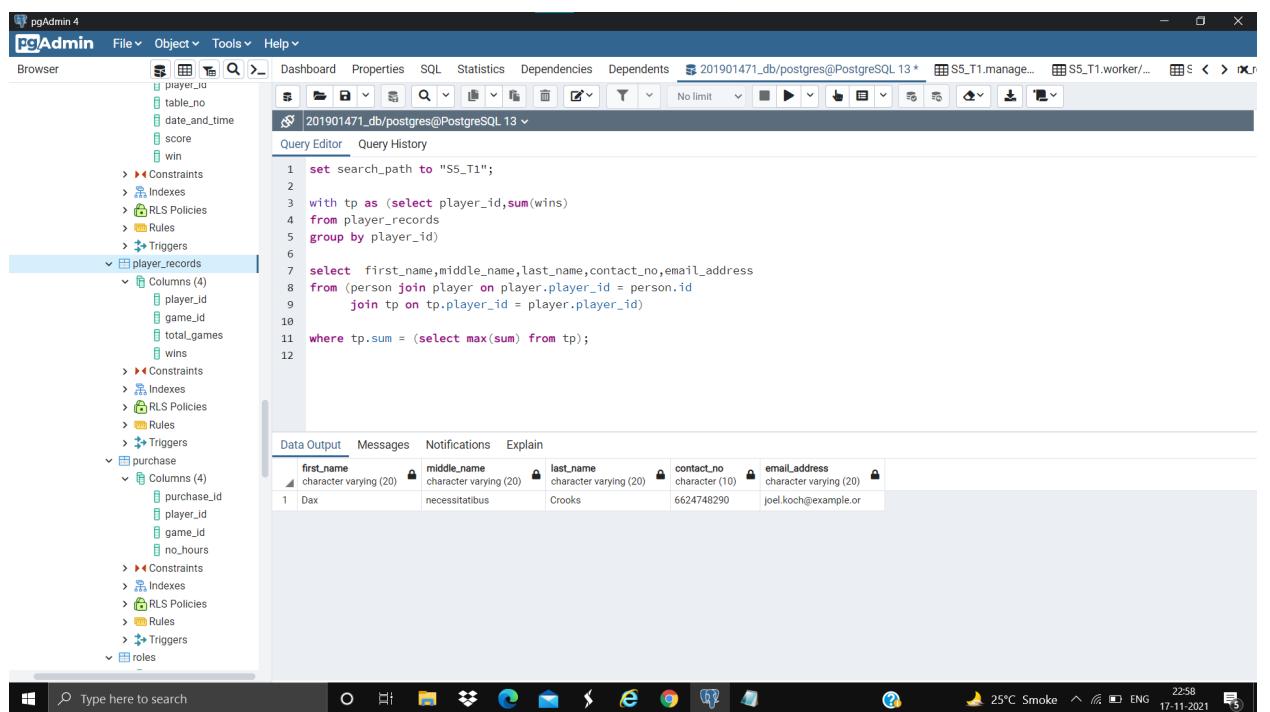
No. of Tuple : 37

## 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);
```



The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows tables like player, table\_no, date\_and\_time, score, win, player\_records, purchase, and roles.
- Query Editor:** Contains the SQL query provided in the text above.
- Data Output:** Displays the result of the query, showing one tuple:

first_name	middle_name	last_name	contact_no	email_address
Dax	necessitatibus	Crooks	6624748290	joel.koch@example.or

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;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like RLS Policies, Rules, Triggers, purchase, team, and transactions. The main area contains the Query Editor with the following SQL code:

```
1 set search_path to gamezone;
2
3 with temp1 as (select account_no,player_id,transactions.type,sum(amount) as amount
4 from transactions natural join account
5 group by account_no,player_id,transactions.type
6 order by account_no asc)
7
8 select person.id,type,amount
9 from temp1 join person on temp1.player_id=person.id;
10
```

The Data Output pane shows the results of the query, which consists of 285 rows. The columns are id (integer), type (character varying(20)), and amount (bigint). The data includes various transaction types (loss, pay, win) and their corresponding amounts for different player IDs.

id	type	amount
263	135	4766
264	135	1622
265	135	4405
266	136	7371
267	136	512
268	137	1562
269	137	2122
270	137	2891
271	138	8951
272	138	4631
273	142	4583
274	142	4390
275	142	1049
276	145	4064
277	145	3803
278	145	3861
279	146	2343
280	146	2302
281	146	3915
282	147	729
283	147	5120
284	149	3206
285	149	4055

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);
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database structure:
  - team\_records
  - transactions
    - Columns (6): account\_no, date\_and\_time, amount, type, method
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - worker
    - Columns (2): worker\_id, role
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
    - Trigger Functions
    - Types
    - Views
  - SM\_DB
  - public
  - sv\_db
  - Subscriptions
  - postgres
  - Login/Group Roles
  - Tablespaces
- Query Editor:** Contains the SQL query:

```
1 set search_path to "S5_T1";  
2  
3 with tp as (select r_name,max(salary/(workhours*30))  
4     from roles  
5     group by r_name)  
6  
7 select person.id,person.first_name,person.last_name,worker.role,tp.max  
8   from ( person join worker on person.id = worker.worker_id  
9          join tp on tp.r_name = worker.role)  
10  
11 where tp.max = (select max(tp.max) from tp);  
12  
13  
14  
15
```

- Data Output:** Displays the results of the query:

	id	first_name	last_name	role	max
1	76	Judy	Ebert	ipsa	150
2	104	Hellen	Morrisette	ipsa	150

- Messages:** Shows a success message: "Successfully run. Total query runtime: 134 msec. 2 rows affected."

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;
```

The screenshot shows the pgAdmin 4 interface with a database browser on the left and a query editor on the right. The query editor contains the two-part SQL statement provided above. The results pane on the right displays the output of the query, which is a table showing the count of players for each game type.

type	no_of_players
Sports	24
Casino	24
Racing	27
Action	11
Board	5
Adventure	9

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;
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects, with 'game\_table' selected. The 'Query Editor' pane contains the following SQL code:

```
1 SET SEARCH_PATH TO GAMEZONE;
2
3
4
5 select game_id, sum(capacity)
6 from game_table
7 group by game_id;
```

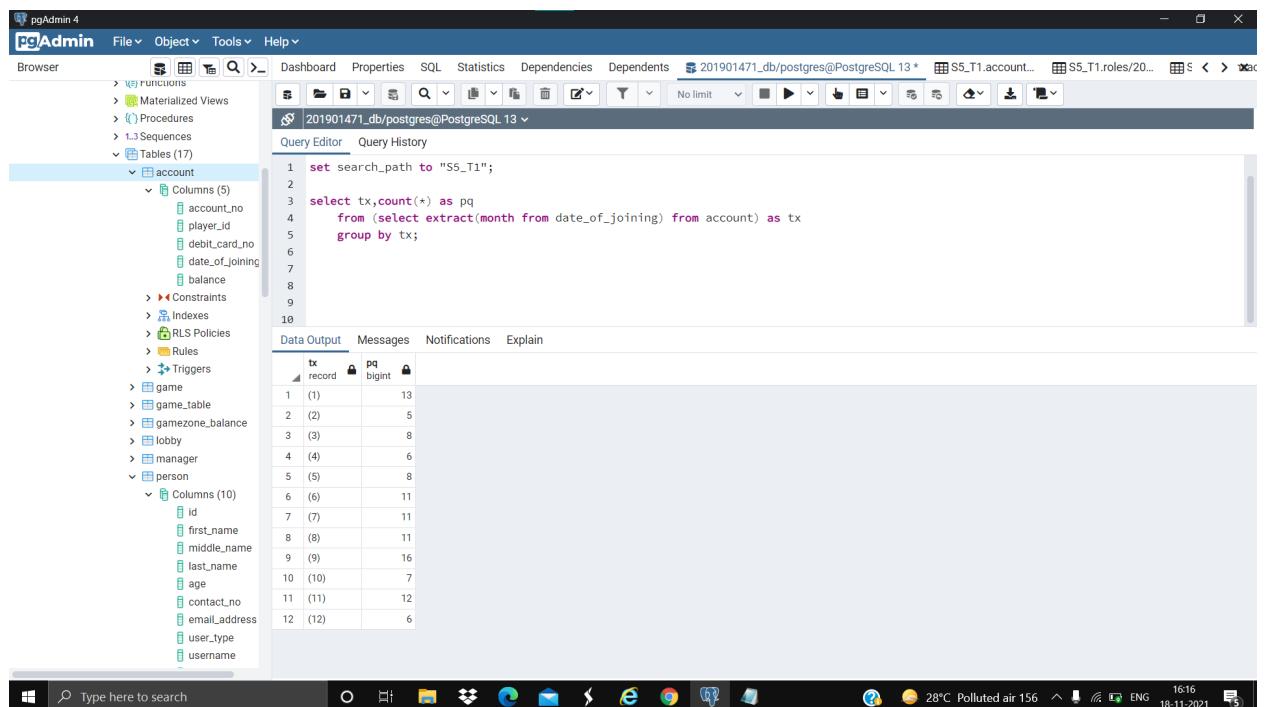
The 'Data Output' pane shows the results of the query:

game_id	sum
21	30
22	21
23	14
24	3
25	17
26	22
27	37
28	20
29	28
30	33
31	13
32	1
33	5
34	18
35	2
36	16
37	27
38	23
39	8
40	11
	34

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;
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
1 set search_path to "S5_T1";  
2  
3 select tx, count(*) as pq  
4   from (select extract(month from date_of_joining) as tx  
5        group by tx;  
6  
7  
8  
9  
10
```

The results table has two columns: 'tx' and 'pq'. The data is:

tx	pq
1 (1)	13
2 (2)	5
3 (3)	8
4 (4)	6
5 (5)	8
6 (6)	11
7 (7)	11
8 (8)	11
9 (9)	16
10 (10)	7
11 (11)	12
12 (12)	6

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;
```

The screenshot shows the pgAdmin 4 interface with a query editor containing the following SQL code:

```

1 set search_path to gamezone;
2
3 with tem as
4 (select t1.player_id,t1.game_id,sum(t2.price)/count(distinct t2.player_id) as cum_sum
5 from (select player_id,game.game_id,price
6      from game,player
7     where type = 'Racing'
8    order by price) as t1
9 inner join
10  (select player_id,game.game_id,price
11    from game,player
12   where type = 'Racing'
13  order by price) as t2
14 on t2.price <= t1.price
15 group by t1.player_id,t1.game_id,t1.price
16 order by cum_sum)
17
18 select account.player_id,count(cum_sum)
19 from account join tem on account.player_id=tem.player_id
20 where cum_sum <= balance
21 group by account.player_id;
22

```

The data output table on the right shows the results of the query, with columns 'player\_id' and 'count'.

player_id	count
95	118
96	119
97	120
98	121
99	123
100	128
101	129
102	130
103	131
104	133
105	134
106	135
107	136
108	137
109	138
110	142
111	145
112	146
113	147
114	149

No. of tuple : 114

### 34. Show Leaderboard of all games.

```
select * from SHOW_PLAYER_LEADERBOARD(0);
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database schema with tables `player` and `player_history`. The `player` table has columns: id, first\_name, middle\_name, last\_name, age, contact\_no, email\_address, user\_type, username, password. The `player_history` table has columns: player\_id, table\_no, date\_and\_time, score, win.
- Query Editor:** Contains the SQL code for the `SHOW_PLAYER_LEADERBOARD` function:

```
1 set search_path to gamezone;
2
3 CREATE OR REPLACE FUNCTION SHOW_PLAYER_LEADERBOARD(GID int)
4 RETURNS TABLE(game_id int,player_id int,max_score int)
5 language 'plpgsql'
6 AS $BODY$
7 BEGIN
8
9 IF (GID=0) THEN
10    RETURN QUERY (SELECT game_table.GAME_ID,PLAYER_ID,MAX(SCORE)
11    FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
12    GROUP BY PLAYER_ID,game_table.GAME_ID
13    ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
14 ELSE
15    RETURN QUERY (SELECT game_table.game_id,PLAYER_ID,MAX(SCORE)
16    FROM PLAYER_HISTORY NATURAL JOIN GAME_TABLE
17    WHERE game_table.GAME_ID=GID
18    GROUP BY PLAYER_ID,game_table.GAME_ID
19    ORDER BY game_table.GAME_ID ASC,MAX(SCORE) DESC);
20 END IF;
21
22 $BODY$;
23
24 select * from SHOW_PLAYER_LEADERBOARD(0);
```

- Data Output:** Displays the results of the query, showing 201 rows with columns: game\_id, player\_id, max\_score.

game_id	player_id	max_score
171	30	89
172	30	51
173	31	10
174	31	45
175	33	50
176	33	16
177	33	18
178	33	52
179	33	23
180	33	107
181	33	30
182	34	128
183	34	14
184	35	39
185	35	11
186	35	21
187	36	21
188	36	45
189	37	3
190	38	50
191	38	22
192	38	27
193	39	129
194	39	73
195	39	105
196	39	147
197	40	12
198	40	112
199	40	138
200	40	68
201	40	46

- Messages:** Shows "Successfully run. Total query runtime: 275 msec. 201 rows affected."

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
```

The screenshot shows the pgAdmin 4 interface. On the left is the 'Browser' pane displaying database schema information for 'gamezone'. In the center is the 'Query Editor' pane containing the SQL query provided above. To the right is the 'Data Output' pane, which displays the results of the query as a table. The table has three columns: game\_id, player\_id, and score. The data consists of 20 rows, with the first few rows being:

game_id	player_id	score
1	1	62
2	1	2
3	2	92
4	2	34
5	2	88
6	3	117
7	3	60
8	3	85
9	4	106
10	4	81
11	5	65
12	5	9
13	5	86
14	6	119
15	6	90
16	6	106
17	7	73
18	7	133
19	7	44
20	8	134

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
```

```
set search_path to gamezone;
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
```

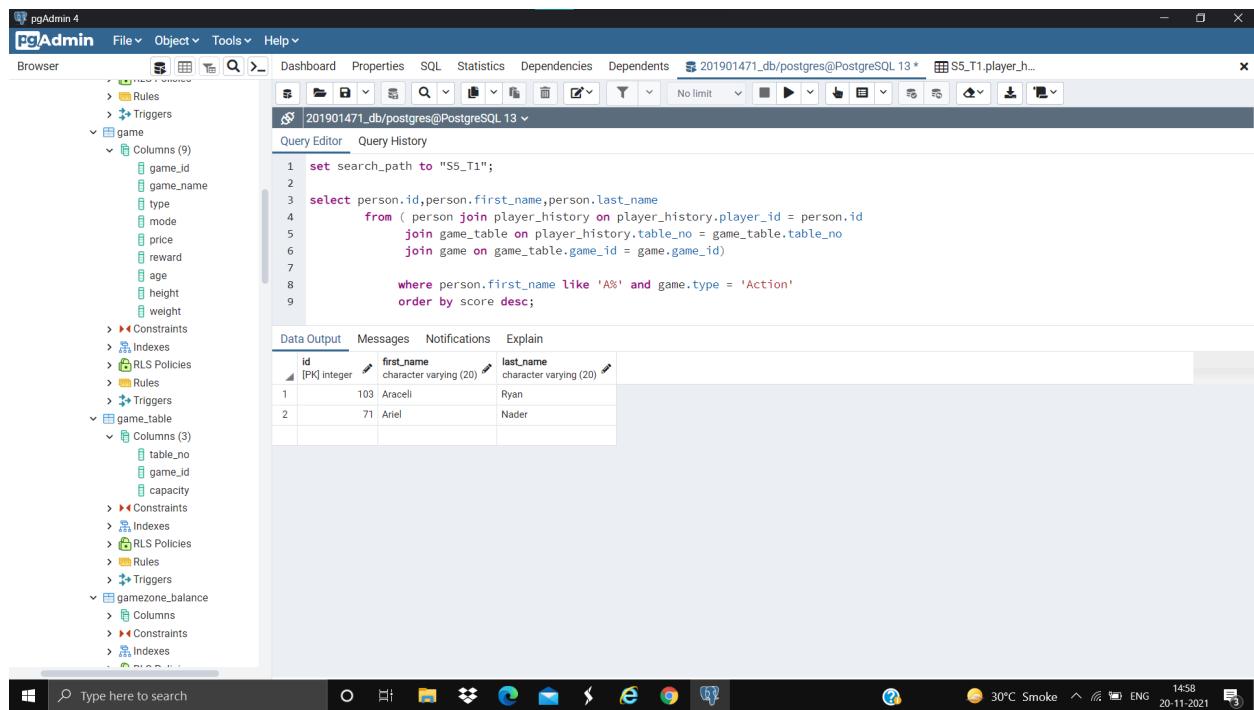
team_id	player_id
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25

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;
```



The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database schema structure:
  - game**: Contains columns (id, game\_name, type, mode, price, reward, age, height, weight) and constraints.
  - game\_table**: Contains columns (table\_no, game\_id, capacity).
  - gamezone\_balance**: Contains columns (id, balance).
- Query Editor:** Displays the SQL query:

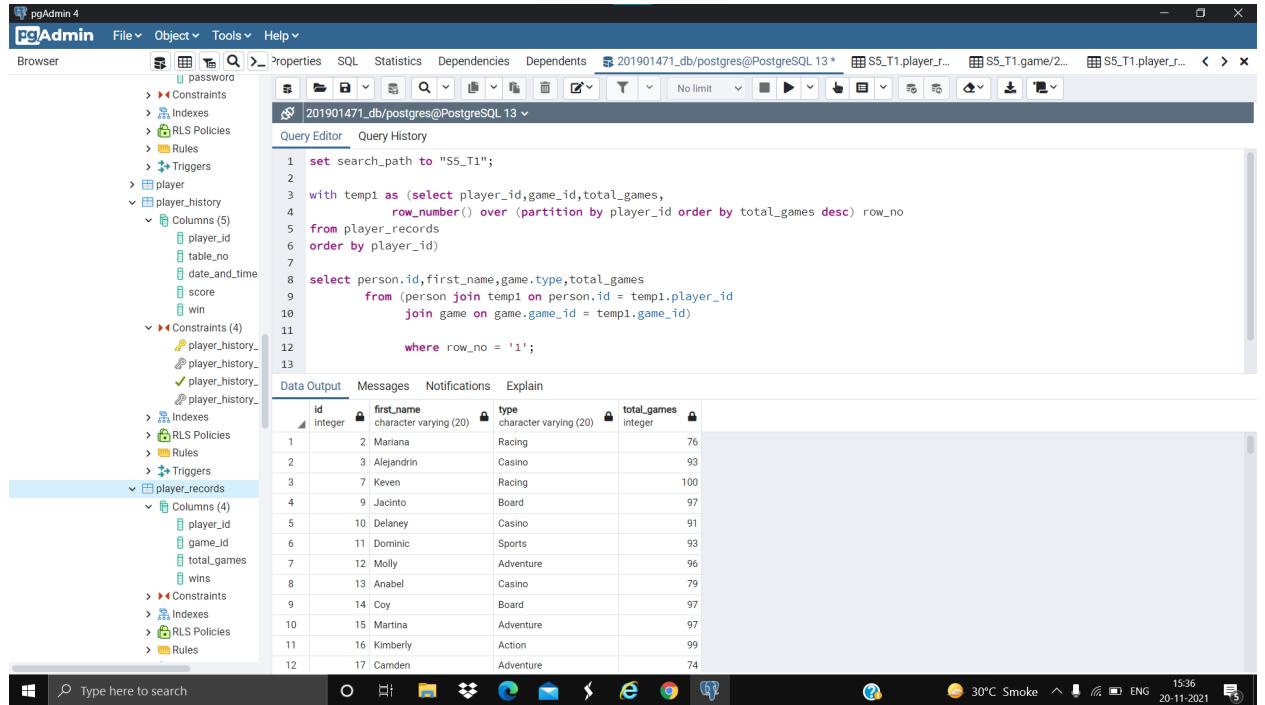
```
1 set search_path to "S5_T1";  
2  
3 select person.id, person.first_name, person.last_name  
      from ( person join player_history on player_history.player_id = person.id  
             join game_table on player_history.table_no = game_table.table_no  
             join game on game_table.game_id = game.game_id)  
where person.first_name like 'A%' and game.type = 'Action'  
order by score desc;
```
- Data Output:** Shows the results of the query:

	id	first_name	last_name
1	103	Arcelli	Ryan
2	71	Ariel	Nader
- Messages:** No messages displayed.
- Notifications:** No notifications displayed.
- Explain:** No explain plan displayed.

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';
```



```
1 set search_path to "S5_T1";  
2  
3 with temp1 as (select player_id,game_id,total_games,  
4                     row_number() over (partition by player_id order by total_games desc) row_no  
5                 from player_records  
6                 order by player_id)  
7  
8 select person.id,first_name,game.type,total_games  
9      from (person join temp1 on person.id = temp1.player_id  
10             join game on game.game_id = temp1.game_id)  
11  
12 where row_no = '1';  
13
```

	id	first_name	type	total_games
1	2	Mariana	Racing	76
2	3	Alejandrín	Casino	93
3	7	Kevan	Racing	100
4	9	Jacinto	Board	97
5	10	Delaney	Casino	91
6	11	Dominic	Sports	93
7	12	Molly	Adventure	96
8	13	Anabel	Casino	79
9	14	Coy	Board	97
10	15	Martina	Adventure	97
11	16	Kimberly	Action	99
12	17	Camden	Adventure	74

No. of tuple : 114

### 39. Show Leaderboard of game\_id 39.

```
select * from SHOW_PLAYER_LEADERBOARD(39);
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database schema structure, including tables like `person`, `player`, and `player_history`, and their respective columns and constraints.
- Query Editor:** Contains the SQL code for the function `SHOW_PLAYER_LEADERBOARD` and the execution command `select * from SHOW_PLAYER_LEADERBOARD(39);`.
- Data Output:** Displays the results of the query as a table:

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

Below the table, the message area indicates: "Successfully run. Total query runtime: 173 msec. 4 rows affected."

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);
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL 13 (S5\_T1)
- Databases:** S5\_T1
- Query Editor:** Contains the following SQL code:

```

1 set search_path to gamezone;
2
3 CREATE OR REPLACE FUNCTION gamezone.show_player_rank(
4     pid integer, gid integer)
5     RETURNS TABLE(player integer, game integer, _score integer, _rank bigint)
6     LANGUAGE 'plpgsql'
7 AS $BODY$
8 BEGIN
9
10    IF (pid=0 and gid=0) then
11        return query (select player_id,game_id,score,
12                        row_number() over (partition by game_id order by score desc) row_no
13                   from player_history natural join game_table
14                   order by player_id,game_id);
15
16    elseif gid=0 then
17        return query (select player_id,game_id,score,row_no from (select player_id,game_id,sc
18                        row_number() over (partition by game_id order by score desc)  row_no
19                     from player_history natural join game_table) as T1 where player_id=pid order by game_
20
21    else
22
23        return query (select player_id,game_id,score,row_no from (select player_id,game_id,sc
24                        row_number() over (partition by game_id order by score desc)  row_no
25                     from player_history natural join game_table) as T1 where player_id=pid and game_id=gid
26
27    end if;
28 END;
29 $BODY$;
30
31 select * from show_player_rank(2,0);

```
- Data Output:** A table showing the results of the query. The columns are player, game, score, and \_rank. The data is as follows:

player	game	score	_rank
1	2	1	1258
2	2	3	100
3	2	17	2343
4	2	18	100
5	2	26	2500
- Messages:** Successfully run. Total query runtime: 59 msec.  
5 rows affected.

No. of tuple: 5

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

```
select * from show_player_rank(2,17);
```

The screenshot shows the pgAdmin 4 interface. On the left is the Object Browser tree, which includes servers, databases (2010\_1429\_db), casts, catalogs, event triggers, extensions, foreign data wrappers, languages, publications, schemas (gamezone, gamezone), functions (get\_player\_profile, get\_team\_profile, show\_player\_leaders), tables (account, account), and constraints. The 'Query Editor' tab is active, showing the following SQL code:

```
1 set search_path to gamezone;
2
3 CREATE OR REPLACE FUNCTION gamezone.show_player_rank(
4     pid integer, gid integer)
5 RETURNS TABLE(player integer, game integer, _score integer, _rank bigint)
6 LANGUAGE 'plpgsql';
7 AS $BODY$
8 BEGIN
9
10    IF (pid=0 and gid=0) then
11        return query (select player_id,game_id,score,
12                        row_number() over (partition by game_id order by score desc) row_no
13                   from player_history natural join game_table
14                  order by player_id,game_id);
15
16    elseif gid=0 then
17        return query (select player_id,game_id,score, row_no from (select player_id,game_id,sc
18                        row_number() over (partition by game_id order by score desc) row_no
19                     from player_history natural join game_table) as T1 where player_id=pid order by game_
20
21    else
22
23        return query (select player_id,game_id,score, row_no from (select player_id,game_id,sc
24                        row_number() over (partition by game_id order by score desc) row_no
25                     from player_history natural join game_table) as T1 where player_id=pid and game_id=gt
26
27    end if;
28
29    $BODY$;
30
31 select * from show_player_rank(2,17);
```

The 'Data Output' tab shows the results of the query:

player	game	_score	_rank
1	2	17	2343

Below the code editor, the message bar says "Successfully run. Total query runtime: 45 msec. 1 rows affected."

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('abcdefghijklmnopqrstuvwxyz0123456789',((random()*(36-1)+1)::integer),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('abcdefghijklmnopqrstuvwxyz0123456789',((random()*(36-1)+1)::integer),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('abcdefghijklmnopqrstuvwxyz0123456789',((random()*(36-1)+1)::integer),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('abcdefghijklmnopqrstuvwxyz0123456789',((random()*(36-1)+1)::integer),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('abcdefghijklmnopqrstuvwxyz0123456789',((random()*(36-1)+1)::integer),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('abcdefghijklmnopqrstuvwxyz0123456789',((random()*(36-1)+1)::integer),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,person  
.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;
$BODY$;
```