DELIVERABLE -3

Queries:

1. Select all the players that lie in the age group of 16 to 25.

```
olympics=# select country_id,player_name,age from participant where age between 16 and 25;
 country_id |      player_name       | age
------------+------------------------+-----
          3 | Adam Tony Forsyth      |  23
          4 | Jaume Fort Mauri       |  22
          5 | Renato Rene Fortaleza  |  17
          7 | Saina Nehwal           |  24
          7 | PV Sindhu              |  21
          8 | Arvo Ossian Aaltonen   |  22
          2 | Stefan Remco Aartsen   |  21
          7 | Neeraj Chopra          |  23
         11 | Peter Heatly           |  24
         12 | Sahar Helal            |  17
          7 | Ravi Kumar             |  23
         13 | Zavur Uguev            |  25
(12 rows)
```

2. Retrieve a list of all players who played in 2020 Olympics:

```
^Colympics=# select distinct p.player_id, p.player_name, h.date from participant as p, held_at as h where cast(date as varchar) like '2020%';
 player_id |         player_name          |    date
-----------+------------------------------+------------
        10 | Einar Ferdinand Aalto        | 2020-01-01
        19 | Sahar Helal                  | 2020-01-01
         6 | Renato Rene Fortaleza        | 2020-01-01
        22 | Sarah Robles                 | 2020-01-01
         3 | Sonia Blanco Bernal          | 2020-01-01
         2 | Christine Jacoba Aaftink     | 2020-01-01
        13 | Paul Kipngetich Tanui        | 2020-01-01
        17 | Johannes Vetter              | 2020-01-01
        15 | Mary Kom                     | 2020-01-01
        16 | Neeraj Chopra                | 2020-01-01
        21 | Zavur Uguev                  | 2020-01-01
         9 | PV Sindhu                    | 2020-01-01
        12 | Stefan Remco Aartsen         | 2020-01-01
         1 | Edgar Lindenau Aabye         | 2020-01-01
        14 | Lee Calhoun                  | 2020-01-01
        11 | Arvo Ossian Aaltonen         | 2020-01-01
         8 | Saina Nehwal                 | 2020-01-01
        20 | Ravi Kumar                   | 2020-01-01
         5 | Jaume Fort Mauri             | 2020-01-01
         4 | Adam Tony Forsyth            | 2020-01-01
         7 | Joseph Cephis Joe Fortenberry | 2020-01-01
        18 | Peter Heatly                 | 2020-01-01
(22 rows)
```

3. Retrieve a list of all sports played at Deodre Aquatics Centre:

```
olympics=# select e.event_name from event as e, venue as v where v.venue_name = 'Deodoro Aquatics Centre';
      event_name
----------------------
 Basketball
 Doubles Badminton
 Boxing
 Freestyle Swimming
 Singles Badminton
 Weight Lifting
 100m Sprint
 110m Hurdles
 Javelin Throw
 Diving
 Synchronised Swimming
 Wrestling
 Breaststroke Swimming
 4X100m Sprint
(14 rows)
```

1)All participants that are female and are from India

```
SELECT * FROM PARTICIPANT AS p,COUNTRY AS c
 WHERE GENDER='F' AND p.country_id=c.country_id
 AND c.country_name='India';
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 player_id | country_id | player_name  | gender | age | weight | country_id | country_name
-----------+------------+--------------+--------+-----+--------+------------+--------------
         8 |          7 | Saina Nehwal | F      |  24 |     67 |          7 | India
         9 |          7 | PV Sindhu    | F      |  21 |     70 |          7 | India
        15 |          7 | Mary Kom     | F      |  34 |     50 |          7 | India
(3 rows)
```

2)List all the countries who took part in badminton

```
SELECT DISTINCT country_name from COUNTRY
INNER JOIN PARTICIPANT ON PARTICIPANT.country_id=COUNTRY.country_id
INNER JOIN COMPETES ON PARTICIPANT.player_id=COMPETES.player_id
INNER JOIN EVENT ON EVENT.event_id=COMPETES.event_id
WHERE EVENT.event_name = 'Boxing';
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 country_name
--------------
 Australia
 India
 Phillipines
(3 rows)
```

3)No of medals won by KENYA

```
SELECT COUNT(*) FROM WINNER AS w, PARTICIPANT as p, COUNTRY AS c
WHERE w.player_id=p.player_id AND p.country_id=c.country_id AND c.country_name='Kenya';
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 count
-------
     2
(1 row)
```

4)NO OF MEDALS WON BY EACH COUNTRY

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 country_name | count
--------------+-------
 India        |     4
 Kenya        |     2
 Germany      |     1
 Denmark      |     1
 Netherlands  |     1
 Spain        |     1
 USA          |     3
(7 rows)
```

```
SELECT c.country_name, COUNT(*) FROM WINNER AS w, PARTICIPANT as p, COUNTRY AS c
WHERE w.player_id=p.player_id AND p.country_id=c.country_id
GROUP BY c.country_name;
```

5)COUNTRIES THAT WON A MEDALS IN 2020 AND ITS COUNT

```
SELECT c.country_name, COUNT(*) FROM WINNER AS w, PARTICIPANT as p, COUNTRY AS c, COMPETES as com
WHERE w.player_id=p.player_id AND p.country_id=c.country_id
 AND com.player_id=p.player_id AND com.year='2020'
GROUP BY c.country_name;
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 country_name | count
--------------+-------
 India        |     2
 Germany      |     1
(2 rows)
```

6)COUNTRIES THAT DID NOT WIN A MEDAL IN 1996

```
(SELECT c.country_name FROM COUNTRY AS c )
EXCEPT
( SELECT c.country_name FROM COUNTRY AS c , WINNER AS w, PARTICIPANT as p, COMPETES as com
WHERE w.player_id=p.player_id AND p.country_id=c.country_id
 AND com.player_id=p.player_id AND com.year='1996'
GROUP BY c.country_name);
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 country_name
---------------
 Egypt
 Phillipines
 India
 Kenya
 Australia
 Germany
 Great Britain
 Russia
 Finland
 Netherlands
 Spain
(11 rows)
```

7)show all the events being held at Panathinaiko Stadium

```
SELECT DISTINCT e.event_name from EVENT as e, VENUE as v ,HELD_AT as h WHERE
h.venue_id=v.venue_id and h.event_id=e.event_id and v.venue_name='Panathinaiko Stadium';
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
  event_name
---------------
 110m Hurdles
 4X100m Sprint
 Javelin Throw
(3 rows)
```

8) show all the countries that have won a gold medal

```
SELECT country_name,COUNT(medal) as GOLD from (SELECT p.player_id, c.country_name ,medal from
 PARTICIPANT as p, COUNTRY AS c,WINNER AS w where w.player_id=p.player_id AND p.country_id=c.country_id AND w.medal='Gold'
group by e1.country_name
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 country_name | gold
--------------+------
 Denmark      |    1
 India        |    1
 Kenya        |    2
 Netherlands  |    1
 USA          |    2
(5 rows)
```

9) SHOW THE NUMBER OF GOLD ,SILVER AND BRONZE WON BY EACH COUNTRY

```sql
SELECT country_name, GOLD, SILVER,BRONZE from(

(
SELECT country_name,COUNT(medal) as GOLD from COUNTRY as c
LEFT OUTER JOIN PARTICIPANT as p ON p.country_id=c.country_id
LEFT OUTER JOIN WINNER AS w ON p.player_id=w.player_id AND w.medal='Gold'
GROUP BY country_name
) as g natural JOIN
(
    SELECT country_name,COUNT(medal) as SILVER from COUNTRY as c
LEFT OUTER JOIN PARTICIPANT as p ON p.country_id=c.country_id
LEFT OUTER JOIN WINNER AS w ON p.player_id=w.player_id AND w.medal='Silver'
GROUP BY country_name

) AS s natural JOIN
(
    SELECT country_name,COUNT(medal) as BRONZE from COUNTRY as c
LEFT OUTER JOIN PARTICIPANT as p ON p.country_id=c.country_id
LEFT OUTER JOIN WINNER AS w ON p.player_id=w.player_id AND w.medal='Bronze'
GROUP BY country_name

) AS b

);
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
 country_name  | gold | silver | bronze
---------------+------+--------+--------
 Egypt         |    0 |      0 |      0
 Phillipines   |    0 |      0 |      0
 India         |    1 |      2 |      1
 Kenya         |    2 |      0 |      0
 Australia     |    0 |      0 |      0
 Germany       |    0 |      1 |      0
 Great Britain |    0 |      0 |      0
 Denmark       |    1 |      0 |      0
 Russia        |    0 |      0 |      0
 Finland       |    0 |      0 |      0
 Netherlands   |    1 |      0 |      0
 Spain         |    0 |      1 |      0
 USA           |    2 |      0 |      1
(13 rows)
```

10) VENUES AT WHICH NO OF EVENTS TAKING PLACE IS GREATER THAN 1

```
SELECT v.venue_name ,count(*) from EVENT as e, VENUE as v ,HELD_AT as h WHERE
h.venue_id=v.venue_id and h.event_id=e.event_id
group by v.venue_name;
```

```
olympics=# \i 'C:/Users/Dell/Documents/dbms_lab/dbms_project/commands.sql'
       venue_name        | count
-------------------------+-------
 Olympic Badminton Arena |     5
 Panathinaiko Stadium    |     5
 Deodoro Aquatics Centre |     6
(3 rows)
```

Performance Analysis:

Select all players who have won gold medals

```
--non-nested
select distinct player_id,player_name from participant as p, winner as w
where w.medal = 'Gold'
and p.player_id=w.player_id;

--nested
select distinct player_id,player_name from participant
where player_id in (
select player_id from winner
where medal = 'Gold');
```

```
olympics=# explain analyze select distinct p. player_id,player_name from participant as p, winner as w where w.medal = 'Gold' and p.player_id=w.player_id;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=40.94..40.98 rows=6 width=122) (actual time=0.083..0.092 rows=5 loops=1)
   ->  Sort  (cost=40.94..40.95 rows=6 width=122) (actual time=0.082..0.085 rows=7 loops=1)
         Sort Key: p.player_id, p.player_name
         Sort Method: quicksort  Memory: 25kB
         ->  Hash Join  (cost=24.20..40.86 rows=6 width=122) (actual time=0.053..0.070 rows=7 loops=1)
               Hash Cond: (p.player_id = w.player_id)
               ->  Seq Scan on participant p  (cost=0.00..14.80 rows=480 width=122) (actual time=0.013..0.018 rows=22 loops=1)
               ->  Hash  (cost=24.12..24.12 rows=6 width=4) (actual time=0.023..0.023 rows=7 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on winner w  (cost=0.00..24.12 rows=6 width=4) (actual time=0.008..0.015 rows=7 loops=1)
                           Filter: ((medal)::text = 'Gold'::text)
                           Rows Removed by Filter: 6
 Planning Time: 0.243 ms
 Execution Time: 0.172 ms
(14 rows)

olympics=# explain analyze select distinct player_id,player_name from participant where player_id in (select player_id from winner where medal = 'Gold');
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=40.40..40.45 rows=6 width=122) (actual time=0.045..0.049 rows=5 loops=1)
   ->  Sort  (cost=40.40..40.42 rows=6 width=122) (actual time=0.045..0.046 rows=5 loops=1)
         Sort Key: participant.player_id, participant.player_name
         Sort Method: quicksort  Memory: 25kB
         ->  Hash Semi Join  (cost=24.20..40.33 rows=6 width=122) (actual time=0.033..0.040 rows=5 loops=1)
               Hash Cond: (participant.player_id = winner.player_id)
               ->  Seq Scan on participant  (cost=0.00..14.80 rows=480 width=122) (actual time=0.007..0.009 rows=22 loops=1)
               ->  Hash  (cost=24.12..24.12 rows=6 width=4) (actual time=0.010..0.010 rows=7 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on winner  (cost=0.00..24.12 rows=6 width=4) (actual time=0.004..0.007 rows=7 loops=1)
                           Filter: ((medal)::text = 'Gold'::text)
                           Rows Removed by Filter: 6
 Planning Time: 0.122 ms
 Execution Time: 0.068 ms
(14 rows)
```

Here, nested query works faster than simple query.

2. Select the countries that have won gold medals:

```
-- select countries that have won gold

-- non-nested
explain analyze
select distinct c.country_id, c.country_name from country as c, participant as p, winner as w
where w.medal = 'Gold'and
c.country_id = p.country_id and
p.player_id = w.player_id;

--

-- nested
explain analyze
select distinct country_id, country_name from country where country_id in (
select p.country_id from participant as p
inner join winner as w
on p.player_id = w.player_id
where w.medal = 'Gold');
```

```
olympics=#
olympics=# explain analyze
olympics-# select distinct c.country_id, c.country_name from country as c, participant as p, winner as w
olympics-# where w.medal = 'Gold'and
olympics-# c.country_id = p.country_id and
olympics-# p.player_id = w.player_id;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=42.54..42.59 rows=6 width=52) (actual time=0.128..0.138 rows=5 loops=1)
   ->  Sort  (cost=42.54..42.56 rows=6 width=52) (actual time=0.127..0.131 rows=7 loops=1)
         Sort Key: c.country_id, c.country_name
         Sort Method: quicksort  Memory: 25kB
         ->  Nested Loop  (cost=24.35..42.47 rows=6 width=52) (actual time=0.073..0.111 rows=7 loops=1)
               ->  Hash Join  (cost=24.20..40.86 rows=6 width=4) (actual time=0.056..0.073 rows=7 loops=1)
                     Hash Cond: (p.player_id = w.player_id)
                     ->  Seq Scan on participant p  (cost=0.00..14.80 rows=480 width=8) (actual time=0.013..0.018 rows=22 loops=1)
                     ->  Hash  (cost=24.12..24.12 rows=6 width=4) (actual time=0.020..0.021 rows=7 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  Seq Scan on winner w  (cost=0.00..24.12 rows=6 width=4) (actual time=0.009..0.015 rows=7 loops=1)
                                 Filter: ((medal)::text = 'Gold'::text)
                                 Rows Removed by Filter: 6
               ->  Index Scan using country_pkey on country c  (cost=0.15..0.27 rows=1 width=52) (actual time=0.004..0.004 rows=1 loops=7)
                     Index Cond: (country_id = p.country_id)
 Planning Time: 0.466 ms
 Execution Time: 0.209 ms
(17 rows)
```

```
olympics=# explain analyze
olympics-# select distinct country_id, country_name from country where country_id in (
olympics(# select p.country_id from participant as p
olympics(# inner join winner as w
olympics(# on p.player_id = w.player_id
olympics(# where w.medal = 'Gold');
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=43.46..43.51 rows=6 width=52) (actual time=0.152..0.161 rows=5 loops=1)
   ->  Sort  (cost=43.46..43.48 rows=6 width=52) (actual time=0.151..0.155 rows=5 loops=1)
         Sort Key: country.country_id, country.country_name
         Sort Method: quicksort  Memory: 25kB
         ->  Nested Loop  (cost=41.02..43.38 rows=6 width=52) (actual time=0.094..0.112 rows=5 loops=1)
               ->  HashAggregate  (cost=40.88..40.94 rows=6 width=4) (actual time=0.073..0.077 rows=5 loops=1)
                     Group Key: p.country_id
                     Batches: 1  Memory Usage: 24kB
                     ->  Hash Join  (cost=24.20..40.86 rows=6 width=4) (actual time=0.048..0.065 rows=7 loops=1)
                           Hash Cond: (p.player_id = w.player_id)
                           ->  Seq Scan on participant p  (cost=0.00..14.80 rows=480 width=8) (actual time=0.014..0.019 rows=22 loops=1)
                           ->  Hash  (cost=24.12..24.12 rows=6 width=4) (actual time=0.020..0.021 rows=7 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                 ->  Seq Scan on winner w  (cost=0.00..24.12 rows=6 width=4) (actual time=0.009..0.015 rows=7 loops=1)
                                       Filter: ((medal)::text = 'Gold'::text)
                                       Rows Removed by Filter: 6
               ->  Index Scan using country_pkey on country  (cost=0.15..0.41 rows=1 width=52) (actual time=0.005..0.005 rows=1 loops=5)
                     Index Cond: (country_id = p.country_id)
 Planning Time: 0.488 ms
 Execution Time: 0.243 ms
(20 rows)
```

Here, simple query works faster than nested query.

MULTIPLE USER:

1. Creation:

```
olympics=# create user audience_member with password 'audience_member' createdb;
CREATE ROLE
olympics=# create user judge with password 'judge' createdb;
CREATE ROLE
olympics=# create user player with password 'player' createdb;
CREATE ROLE
olympics=# create user IOC with password 'IOC' createdb;
CREATE ROLE
```

2. Granting privileges and permissions:

```
olympics=# GRANT pg_read_all_data TO audience_member;
GRANT ROLE
olympics=# grant select ON participant, penalty, winner to judge;
GRANT
```

```
olympics=# grant update(penalty_id,type) on penalty to judge;
GRANT
olympics=# grant select on event, venue, held_at, equipment to player;
GRANT
olympics=# grant select, insert, update on equipment to IOC;
GRANT
```

3. Accessing the database from another user:

```
C:\Program Files\PostgreSQL\14\bin>psql -U judge -d olympics
Password for user judge:
psql (14.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

olympics=> select * from winner;
 player_id | event_id | year | medal
-----------+----------+------+--------
         1 |        1 | 1996 | Gold
         2 |        2 | 2004 | Gold
         5 |        4 | 1992 | Silver
         7 |        1 | 1996 | Bronze
        13 |        7 | 2016 | Gold
        13 |       14 | 2016 | Gold
        14 |        8 | 1992 | Gold
        14 |        8 | 1996 | Gold
        16 |        9 | 2020 | Gold
        17 |        9 | 2020 | Silver
         8 |        5 | 2016 | Silver
         8 |        5 | 2012 | Bronze
         9 |        5 | 2020 | Silver
(13 rows)
```

4. Revoking permission from user

```
C:\Program Files\PostgreSQL\14\bin>psql -U postgres
Password for user postgres:
psql (14.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# revoke select on winner from judge;
ERROR:  relation "winner" does not exist
postgres=# \c olympics
You are now connected to database "olympics" as user "postgres".
olympics=# revoke select on winner from judge;
REVOKE
olympics=# \q

C:\Program Files\PostgreSQL\14\bin>psql -U judge -d olympics
Password for user judge:
psql (14.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

olympics=> select * from winner;
ERROR:  permission denied for table winner
olympics=>
```

Concurrency using transaction program:

1. Begin a transaction program

```
olympics=# begin;
BEGIN
olympics=*# update participant set age= 50;
UPDATE 22
olympics=*#
```

2. Try to concurrently update the same table:

Hence, we can see that while the table is being updated in the transaction program, aother user cannot update it.

```
postgres=# \c olympics
You are now connected to database "olympics" as user "postgres".
olympics=# update participant set weight = 20 where player_id = 1;
```