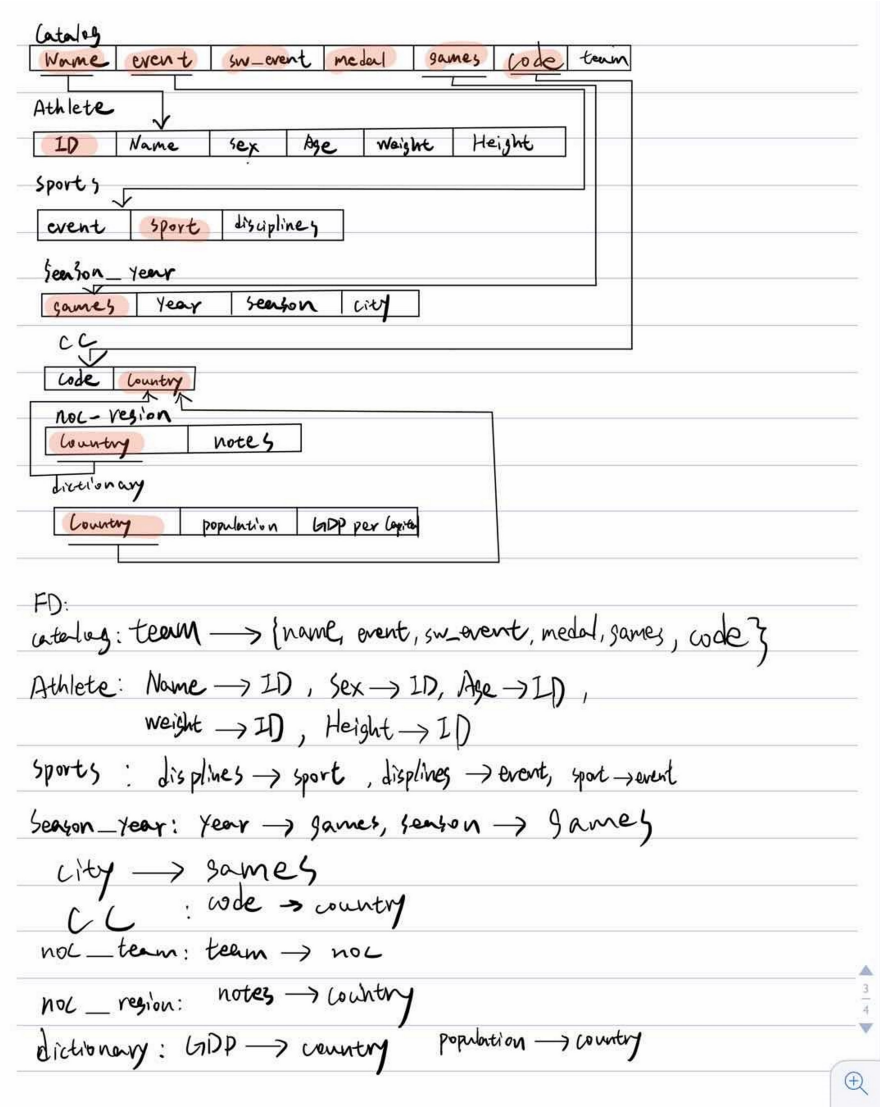
postgreSQL: Analysis of Olympics since 1896

Get all tables from the two web sites below:

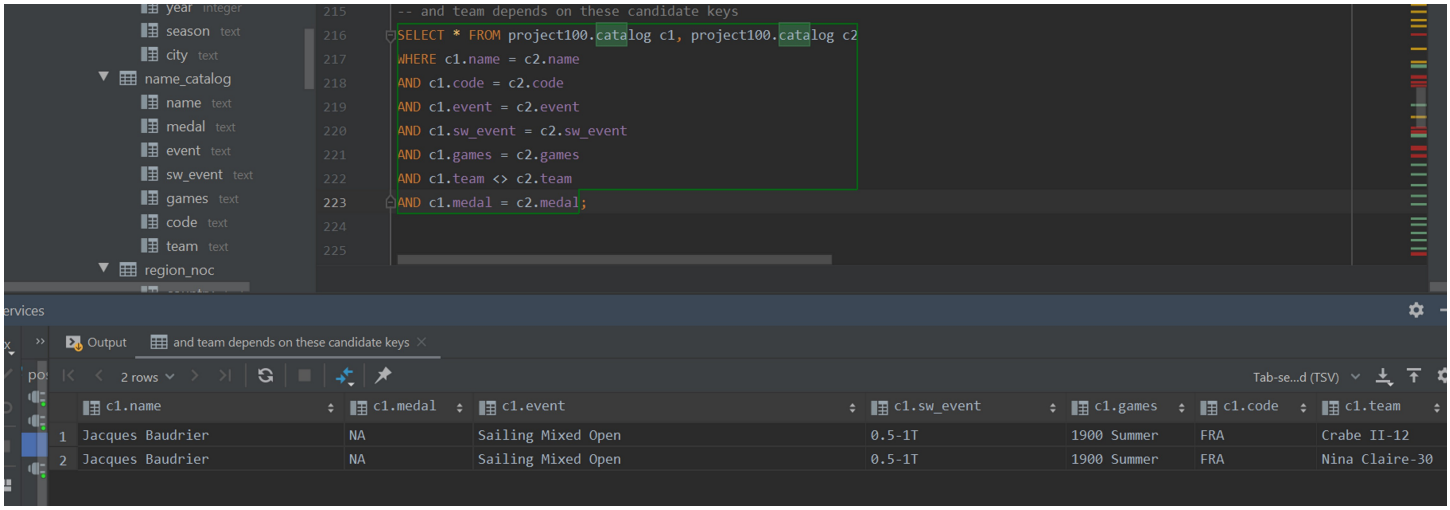
https://www.kaggle.com/the-guardian/olympic-games (Links to an external site.)
(Links to an external site.)https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results (Links to an external site.)

Project Tasks and Deliverables:

1. Combine the given data tables to create a new schema with tables satisfying the third normal form



As shown below, team depends on name, event, sw_event, medal, games, code



2. You should also show the steps of arriving at the 3NF form starting from the original data

1) Add attributes season and games to tables summer and winter

```
-- create a table that combine summer and winter
Alter Table project100.winter
Add season varchar(10);
update project100.winter set season = 'Winter' where season is null;
Alter Table project100.summer
Add season varchar(10);
update project100.summer set season = 'Summer' where season is null;
Alter Table project100.winter
Add games varchar(30);
update project100.winter set games = concat(Cast(year as varchar(10)) ,
Alter Table project100.summer
Add games varchar(30);
update project100.summer set games = concat(Cast(year as varchar(10)) ,
```

2) Create a table that concatenate summer and winter

```
CREATE TABLE project100.summer_winter1 AS
    SELECT *
FROM project100.summer
UNION
SELECT *
FROM project100.winter;
```

3) add first name and last name to summer_winter, and change the values for gender

```sql
ALTER TABLE project100.summer_winter1
ADD COLUMN last_name VARCHAR,
ADD COLUMN first_name VARCHAR,
ADD COLUMN middle_name VARCHAR;
UPDATE project100.summer_winter1
SET last_name = Lower(CASE
WHEN athlete like '% %'
THEN SPLIT_PART(athlete, ' ', 1)
ELSE 'None'
END);
UPDATE project100.summer_winter1
SET first_name = lower(CASE
    WHEN athlete like '% %'
        THEN split_part(athlete, ' ', 2)
ELSE athlete
END);
UPDATE project100.summer_winter1
SET athlete = REPLACE(athlete, ',', '')
, gender = CASE
    WHEN gender = 'Men' THEN 'M'
ELSE 'F'
END;
```

4) add first name and last name to athlete event

```sql
-- split the column name to first name and add them to athlete event
UPDATE project100.athlete_events
SET first_name = CASE
WHEN name like '% %'
THEN LOWER(SPLIT_PART(name, ' ', 1))
ELSE 'None'
END;

-- add a new column last_name
ALTER TABLE project100.athlete_events
ADD COLUMN last_name VARCHAR;

-- split the column name to last name and add them to athlete event
UPDATE project100.athlete_events
SET last_name = LOWER(SUBSTRING(name FROM '\S+$'));

-- add two columns first and last name
ALTER TABLE project100.summer_winter
ADD COLUMN last_name VARCHAR,
ADD COLUMN first_name VARCHAR,
ADD COLUMN middle_name VARCHAR;
COMMIT TRANSACTION;
```

5)merge table summer_winter and athlete_event based on first_name and last_name

```sql
-- merge tables summer winter and athlete events
CREATE TABLE project100.combined8 AS
SELECT  a.id, a.sex, a.age, a.height, a.weight, a.team, a.code, a.games,
       a.year, a.season, a.city, a.sport, a.event, a.medal, a.name,
       sw.season as sw_season, sw.games as sw_games, sw.country as sw_code, sw.gender as sw_sex,
       sw.city as sw_city, sw.medal as sw_medal, sw.athlete as sw_name, sw.sport as sw_sport,
       sw.year as sw_year,sw.event as sw_event,sw.discipline
FROM project100.athlete_events AS a
        FULL OUTER JOIN project100.summer_winter1 AS sw
            ON (a.first_name = sw.first_name OR sw.first_name = Null)
            AND (a.last_name = sw.last_name OR sw.last_name = Null)
            AND a.code= sw.country
            And a.year = sw.year;
update project100.combined8 set name = sw_name where name is null;
update project100.combined8 set sport = sw_sport where sport is null;
update project100.combined8 set year = sw_year where year is null;
update project100.combined8 set medal = sw_medal where medal is null;
update project100.combined8 set code = sw_code where code is null;
update project100.combined8 set city = sw_city where city is null;
update project100.combined8 set season = sw_season where season is null;
update project100.combined8 set games = sw_games where games is null;
update project100.combined8 set sex = sw_sex where sex is null;
COMMIT TRANSACTION;
```

6)Split the combined table into 3NF tables

```sql
-- catalog: name, event, sw_event, medal, games and teams
BEGIN TRANSACTION;
CREATE TABLE project100.catalog AS
Select name,medal, event,sw_event, games,code
From project100.combined8;
COMMIT TRANSACTION;

--  sports contains event, sports, discipline
CREATE TABLE project100.sport AS
Select event,sport,discipline
from project100.combined8;
COMMIT TRANSACTION;


-- rename noc_regions column name
ALTER TABLE project100.noc_regions
RENAME noc TO code;
ALTER TABLE project100.noc_regions
RENAME region TO country;
COMMIT TRANSACTION;
```

```sql
-- concatenate dictionary and noc_regions on noc and region
CREATE TABLE project100.code_country AS
select code, country from project100.noc_regions
union
select code, country from project100.dictionary ;
COMMIT TRANSACTION;

-- drop columns of code from both dictionary and noc regions
ALTER TABLE project100.dictionary
DROP COLUMN code;
ALTER TABLE project100.noc_regions
DROP COLUMN code;
COMMIT TRANSACTION;
```

```sql
-- create a table called athlete
CREATE TABLE project100.athletes AS
    SELECT id, name, sex, age, height, weight
        FROM project100.combined8;
COMMIT TRANSACTION;

-- create a table called season year
CREATE TABLE project100.game_city_year_season AS
select games,year, season, city from project100.combined8;
COMMIT TRANSACTION;
```

```sql
-- create a table called team_noc,
CREATE TABLE project100.team_noc AS
    SELECT code, team
        FROM project100.combined8;
COMMIT TRANSACTION;
```

```sql
-- add a column of identity if it is 1 then the code has occurred for once if it is 2,
-- then the code has occurred twice
ALTER TABLE "100_new".country_code
ADD COLUMN identity int;
INSERT INTO "100_new".country_code (identity) VALUES (ROW_NUMBER() OVER
    (PARTITION BY code ORDER BY country));

CREATE TABLE project100.cc AS
SELECT code, country, ROW_NUMBER() OVER (PARTITION BY code ORDER BY country)
FROM "100_new".country_code;

-- identity checks if one code has occurred more than once
UPDATE "100_new".country_code
SET identity = row_number
FROM project100.occur;
```

1. This schema will be implemented in postgreSQL and the following queries should be answered.
    a. For the year 1992 in Barcelona display the country name and the total number of competitors from that country, including those countries that have no competitors, in

descending order of the number of competitors. Remember not all countries participate in every Olympics game. (15% queries are of this type)

i. SQL code for the query

```
(SELECT zo.*
    FROM project100.zero_occur zo, "100_new".cc cc
    WHERE zo.country <> cc.country)
UNION
(SELECT country, COUNT (*) as counting
FROM (SELECT DISTINCT name, code,games FROM "100_new".name_catalog) nc,
     "100_new".cc,
     "100_new".game_city_year_season gcys
WHERE nc.code = cc.code
AND gcys.year = 1992
AND SUBSTRING(nc.games FROM '^[0-9]{4}') = '1992'
AND gcys.city = 'Barcelona'
AND cc.row_number = 1
GROUP BY cc.country)
ORDER BY count DESC;
```

ii. Number of rows in the result

376

iii. Screenshot of first 10 rows of the result

376 rows | Tx: Auto

	country	count
1	Russia	800
2	United States	775
3	Germany	642
4	Spain	505
5	France	476
6	UK	432
7	Canada	426
8	Italy	420
9	China	384
10	Japan	320

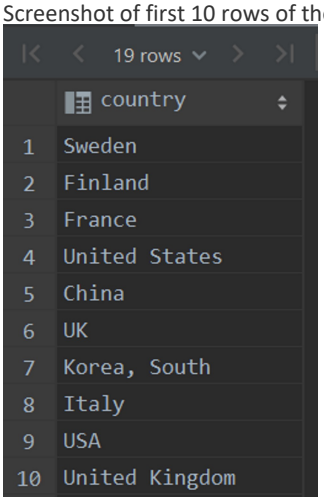
b. For the Vancouver (2010) games, list the competitor countries in the Curling competition. (10% queries are of this type)

i. SQL code for the query

```
-- question 2
--  For the Vancouver (2010) games, list the competitor countries in the Curling competition.
--  (10% queries are of this type)
SELECT DISTINCT cc.country
FROM "100_new".name_catalog nc,
     "100_new".cc,
     "100_new".game_city_year_season gcys,
     "100_new".sport_event se
WHERE (se.sport = 'Curling'
OR nc.sw_event = 'Curling')
AND nc.event = se.event
AND nc.code = cc.code
AND gcys.year = 2010
AND gcys.city = 'Vancouver';
```

ii. Number of rows in the result: 19

iii. Screenshot of first 10 rows of the result



19 rows

	country
1	Sweden
2	Finland
3	France
4	United States
5	China
6	UK
7	Korea, South
8	Italy
9	USA
10	United Kingdom

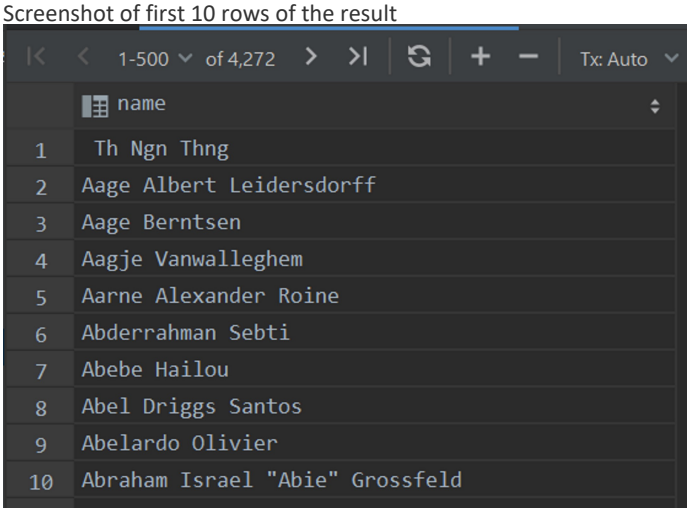
c. List all the competitors who have competed in more than 4 events in any Olympics since 1900. (10% queries are of this type)

i. SQL code for the query

```sql
-- question 3
-- List all the competitors who have competed in more than 4 events in any Olympics since 1900.
-- (10% queries are of this type)
SELECT nc.name
FROM "100_new".name_catalog nc, "100_new".game_city_year_season gcys
WHERE nc.games = gcys.games
AND gcys.year >= 1900
GROUP BY nc.name
HAVING COUNT(DISTINCT event) > 4;
```

ii. Number of rows in the result: 4272

iii. Screenshot of first 10 rows of the result



1-500 of 4,272 Tx: Auto

	name
1	Th Ngn Thng
2	Aage Albert Leidersdorff
3	Aage Berntsen
4	Aagje Vanwalleghem
5	Aarne Alexander Roine
6	Abderrahman Sebti
7	Abebe Hailou
8	Abel Driggs Santos
9	Abelardo Olivier
10	Abraham Israel "Abie" Grossfeld

d. Find the number of competitors in each Olympics game who have competed in at least 3 events and group them by year, after 1940. (15% queries are of this type)

i. SQL code for the query

```sql
-- question 4
-- Find the number of competitors in each Olympics game who have competed in at least 3 events
-- and group them by year, after 1940.
-- (15% queries are of this type)
SELECT COUNT(*), year
FROM(
    SELECT nc.name, year, gcys.games
    FROM "100_new".name_catalog nc, "100_new".game_city_year_season gcys
    Where nc.games = gcys.games
    AND gcys.year > 1940
    GROUP BY nc.name, year, gcys.games
    HAVING COUNT(DISTINCT event) >= 3) AS count
GROUP BY year
ORDER BY year;
```

ii. Number of rows in the result: 24

iii. Screenshot of first 10 rows of the result

Output | postgres.100_new.team_catalog

24 rows

	count	year
1	363	1948
2	625	1952
3	382	1956
4	530	1960
5	588	1964
6	740	1968
7	679	1972
8	605	1976
9	479	1980
10	654	1984

e. Count the number of competitors who were from India in every Olympics held since 1947. (5% queries are of this type)

i. SQL code for the query

```sql
-- question 5
--  Count the number of competitors who were from India in every Olympics held since 1947.
--  (5% queries are of this type)
SELECT COUNT(DISTINCT name)
FROM (SELECT DISTINCT name,code,games FROM "100_new".name_catalog) nc,
     (SELECT DISTINCT year,games FROm "100_new".game_city_year_season) as gcys,
     (SELECT DISTINCT code,country FROM "100_new".cc) cc
WHERE nc.games = gcys.games
AND gcys.year >= 1947
AND nc.code = cc.code
AND cc.country = 'India';
```

ii. Number of rows in the result: 1

iii. Screenshot of first 10 rows of the result

1 row

	count
1	756

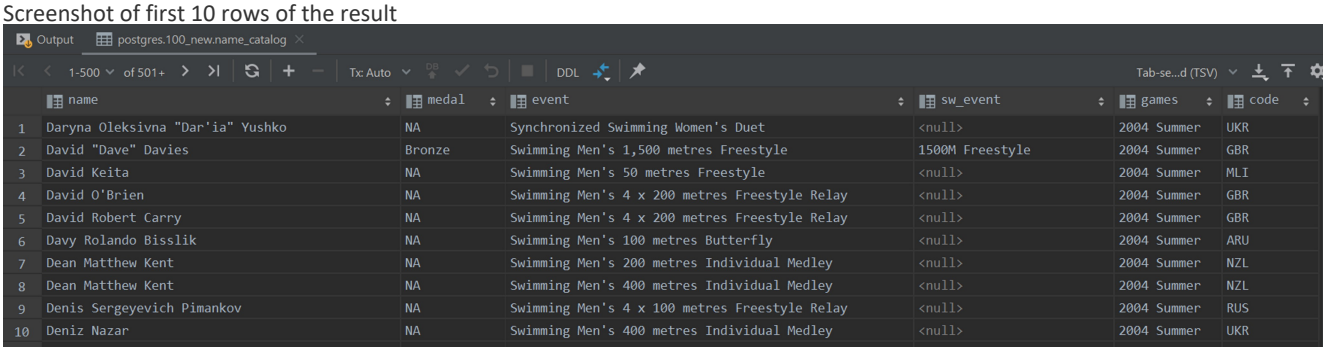
f. Display the results for swimming events (both men and women) in the 2004 Olympics. (10% queries are of this type)

i. SQL code for the query

```sql
-- question 6
--  Display the results for swimming events (both men and women)
--  in the 2004 Olympics.
--  (10% queries are of this type)
SELECT *
FROM "100_new".team_catalog tc
WHERE tc.games LIKE '%2004%'
AND (tc.event like '%Swim%' or tc.event like '%swim%');
```

ii. Number of rows in the result: 1970

iii. Screenshot of first 10 rows of the result



Output | postgres.100_new.name_catalog

1-500 of 501+ | Tx: Auto | DDL | Tab-se...d (TSV)

	name	medal	event	sw_event	games	code
1	Daryna Oleksivna "Dar'ia" Yushko	NA	Synchronized Swimming Women's Duet	<null>	2004 Summer	UKR
2	David "Dave" Davies	Bronze	Swimming Men's 1,500 metres Freestyle	1500M Freestyle	2004 Summer	GBR
3	David Keita	NA	Swimming Men's 50 metres Freestyle	<null>	2004 Summer	MLI
4	David O'Brien	NA	Swimming Men's 4 x 200 metres Freestyle Relay	<null>	2004 Summer	GBR
5	David Robert Carry	NA	Swimming Men's 4 x 200 metres Freestyle Relay	<null>	2004 Summer	GBR
6	Davy Rolando Bisslik	NA	Swimming Men's 100 metres Butterfly	<null>	2004 Summer	ARU
7	Dean Matthew Kent	NA	Swimming Men's 200 metres Individual Medley	<null>	2004 Summer	NZL
8	Dean Matthew Kent	NA	Swimming Men's 400 metres Individual Medley	<null>	2004 Summer	NZL
9	Denis Sergeyevich Pimankov	NA	Swimming Men's 4 x 100 metres Freestyle Relay	<null>	2004 Summer	RUS
10	Deniz Nazar	NA	Swimming Men's 400 metres Individual Medley	<null>	2004 Summer	UKR

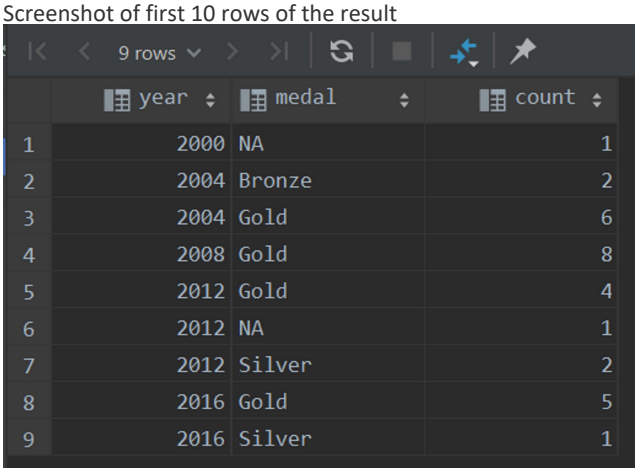
g. Find the medal counts for Michael Phelps across multiple years, show it by year and medal type. (10% queries are of this type)

i. SQL code for the query

```sql
-- question 7
-- Find the medal counts for Michael Phelps across multiple years,
-- show it by year and medal type.
-- (10% queries are of this type)
SELECT gcys.year, nc.medal,count(*)
FROM (SELECT DISTINCT name,event, games, medal FROM "100_new".name_catalog) nc,
     (SELECT DISTINCT games,year from "100_new".game_city_year_season) gcys
Where (nc.name like '%Michael Fred Phelps%')
AND nc.medal is not Null
AND nc.games = gcys.games
Group By gcys.year,nc.medal;
```

ii. Number of rows in the result: 9

iii. Screenshot of first 10 rows of the result



9 rows

	year	medal	count
1	2000	NA	1
2	2004	Bronze	2
3	2004	Gold	6
4	2008	Gold	8
5	2012	Gold	4
6	2012	NA	1
7	2012	Silver	2
8	2016	Gold	5
9	2016	Silver	1

h. Which country has won the most Gold medals in the Men's marathon? (15% queries are of this type)

i. SQL code for the query

```sql
-- question 8
--  Which country has won the most Gold medals in the Men's marathon?
--  (15% queries are of this type)
SELECT nc.code, COUNT(*)
FROM "100_new".name_catalog nc, (SELECT DISTINCT name,sex From "100_new".athletes) a
WHERE (nc.event LIKE '%arathon%'
  or nc.sw_event LIKE '%arathon%')
AND nc.name = a.name
AND (a.sex = 'Men' or a.sex = 'M')
AND nc.medal = 'Gold'
GROUP BY nc.code
ORDER BY COUNT(*) DESC;
```

ii. Number of rows in the result: 18

iii. Screenshot of first 10 rows of the result

18 rows

	code	count
1	ETH	6
2	FRA	5
3	ARG	4
4	FIN	4
5	GDR	4
6	USA	4
7	KEN	3
8	JPN	2
9	RSA	2
10	TCH	2

i. Find the names of competitors who have improved or maintained their standing across three games. Use years after 1940. (10% queries are of this type)

i. SQL code for the query

```sql
UPDATE "100_new".name_catalog
SET number = CASE
WHEN medal = 'Gold'
THEN 3
WHEN medal = 'Silver'
THEN 2
WHEn medal = 'Bronze'
Then 1
END;
```

```sql
SELECT DISTINCT  lower(nc1.name)
FROM "100_new".name_catalog nc1,"100_new".name_catalog nc2,"100_new".name_catalog nc3,
    (SELECT DISTINCT games,year From "100_new".game_city_year_season) gcys
WHEre nc1.medal <> 'NA'
AND nc2.medal <> 'NA'
AND nc3.medal <> 'NA'
AND year > 1940
AND nc1.name = nc2.name
AND nc2.name = nc3.name
AND nc1.number <= nc2.number
AND nc2.number <= nc3.number
AND nc1.games = gcys.games
AND (nc2.games LIKE concat('%',(year + 4),'%'))
AND (nc3.games LIKE concat('%',(year + 8) ,'%'));
```

ii. Number of rows in the result: 613

iii. Screenshot of first 10 rows of the result