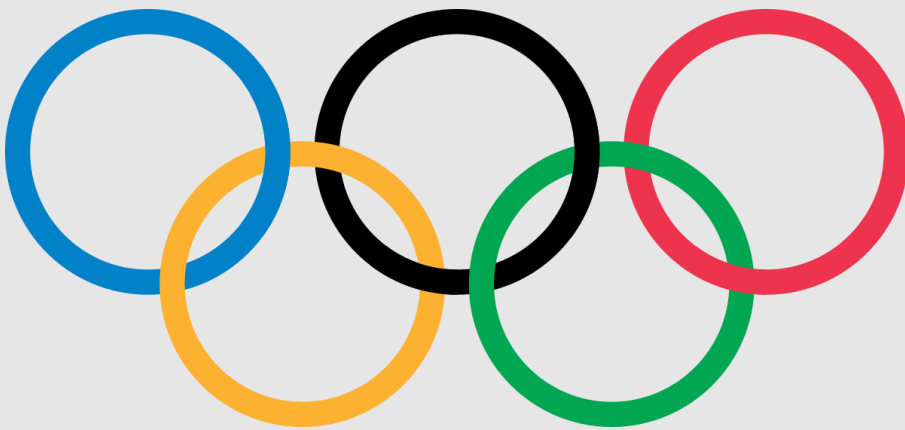


Analysis of Olympics Dataset in SQL



About the Project

We have a dataset containing Information about The Olympic Games Particularly the 2014 Winter Olympics and 2016 Summer Olympics. We will treat the games as a whole for our analysis.

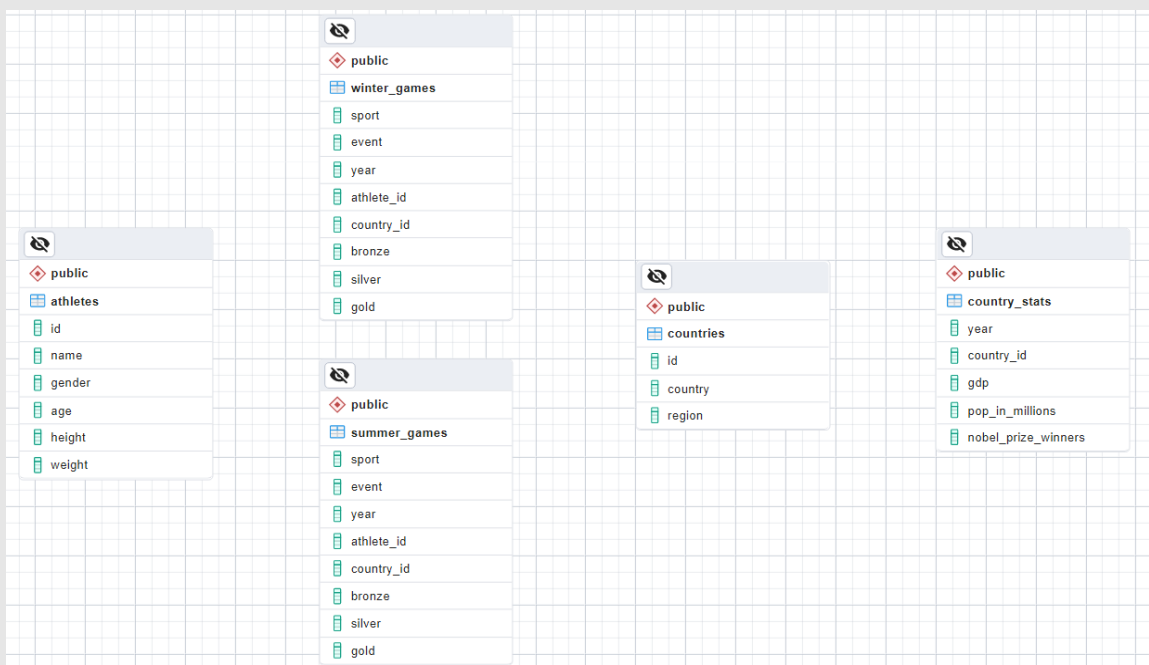
We will analyze and query the database across athlete ,events and country level to uncover patterns and key insights.

We will also try to see trends in performance of countries as a group rather than Individual performance to see whether external factors such as population or per capita income have a key role on performance.

TOOLS :-

This is a SQL Project we will primarily use SQL for this project . However we will use the output as csv from the queries to generate supporting visuals in Power BI when required.

Entity Relationship Diagram



Key Insights

- There is no age limit imposed on athletes as a result the youngest athlete was only 13 years old and the oldest 55.
- The summer games are much more popular with athlete, countries participating and events being 3 times as large as winter games.
- No noticeable gender bias is observed whether it is in athlete representation , share of medals or the number of events by gender. All parameter are equally shared between male and females.
- Athletes between 20 and 30 years of age secured 70% of the total medals awarded.
- 6 out of 10 top performers are from the United States, Michael Phelps being No 1.
- Both high GDP and high GDP per capita level countries are associated with more than 75% of the medal count.
- USA is best performing country by total medals won and also the country with highest representation.

Data Preparation

1. We will check data types of columns

```
column data types

SELECT column_name,data_type
FROM information_schema.columns
WHERE table_schema = 'public'
      AND TABLE_NAME = 'athletes';
```

	column_name name	data_type character varying
1	id	integer
2	name	character varying
3	gender	character varying
4	age	integer
5	height	integer
6	weight	integer

2. Similarly we will check for other tables

```
column data types

SELECT column_name,data_type
FROM information_schema.columns
WHERE table_schema = 'public'
      AND TABLE_NAME = 'country_stats';
```

	column_name name	data_type character varying
1	year	character varying
2	country_id	integer
3	gdp	double precision
4	pop_in_millions	character varying
5	nobel_prize_winners	integer

We find that the data type for year and population columns are incorrect.

3. Correcting the datatype

```
change data types

ALTER TABLE country_stats
ALTER COLUMN year TYPE date USING year::date,
ALTER COLUMN pop_in_millions TYPE numeric(10, 6)
      USING pop_in_millions::numeric(10, 6);
```

	column_name name	data_type character varying
1	year	date
2	country_id	integer
3	gdp	double precision
4	pop_in_millions	numeric
5	nobel_prize_winners	integer

Now the columns have consistent data types , we will combine the summer and winter games into one table.

4. Creating a single combined table for summer and winter games.

```
combined games table

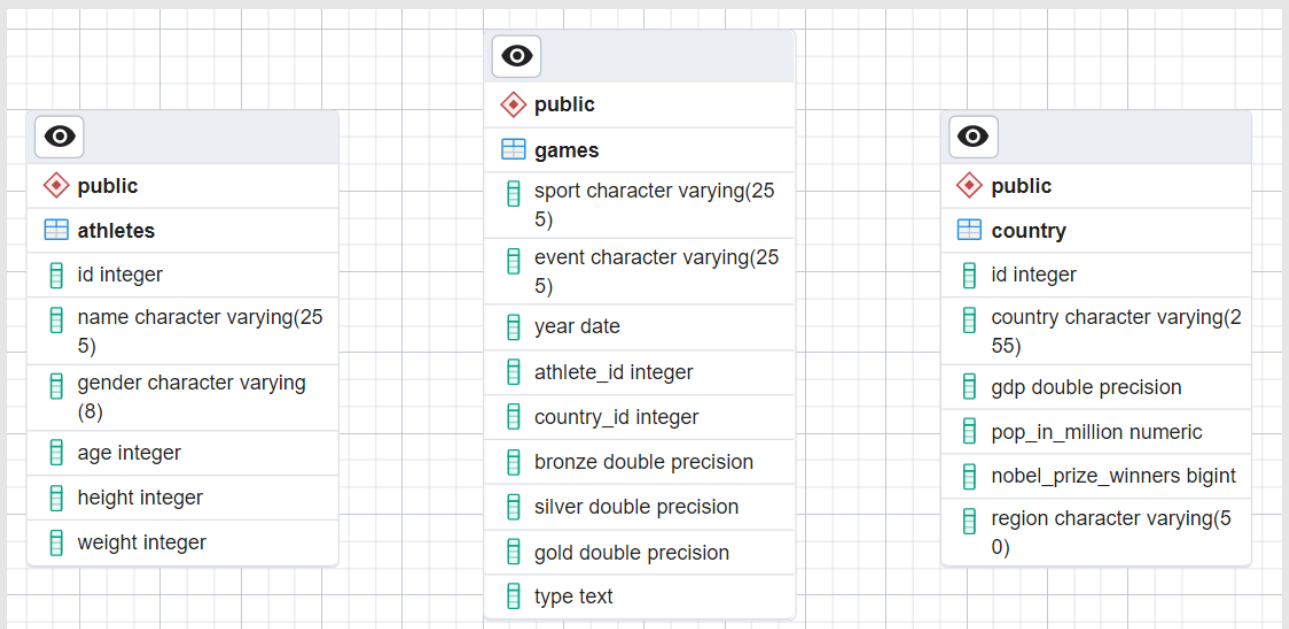
CREATE TABLE games AS
  (SELECT *, 'summer_games' AS type
   FROM summer_games
   UNION ALL
   SELECT *, 'winter_games' AS type
   FROM winter_games)
```

5. Now we will combine the countries table and country stats table and summarize the values in the stats table so we have a single country table for country level analysis.

```
country table

CREATE TABLE country AS
(SELECT country_id AS id,
       country AS name,
       round(avg(coalesce(gdp, 0))) AS gdp,
       round(avg(coalesce(pop_in_millions, 0)),6) pop_in_million,
       sum(nobel_prize_winners) AS nobel_prize_winners,
       region
FROM country_stats
JOIN countries ON country_stats.country_id = countries.id
WHERE extract(YEAR FROM YEAR) >= 2010
GROUP BY country_id, country, region
ORDER BY country_id);
```

6. Now the updated Entity Relationship Diagram



Our data is ready for analysis. We will deal with null and missing values during our analysis

Exploratory Data Analysis

EDA

```
-- Number of countries participating

SELECT COUNT(DISTINCT name)
FROM country

-- Number of athletes

SELECT COUNT(DISTINCT id)
FROM athletes;





-- Count of category and events

SELECT COUNT(DISTINCT sport) AS category,
       COUNT(DISTINCT event) AS sports
FROM games;
```

4215 athletes from
203 countries took
part in
127 events across
6 categories

Count of Medals

```
SELECT COUNT(gold) AS gold,
       COUNT(silver) AS silver,
       COUNT(bronze) AS bronze,
       SUM(gold)+SUM(silver)+SUM(bronze) AS total
FROM games;
```

gold bigint 	silver bigint 	bronze bigint 	total double precision 
206	194	188	588

Age range of Athletes

```
SELECT min(age),
       max(age)
FROM athletes
```

Age range between
13 and 55 confirms
no age limit criteria
imposed by
The Olympics

Athlete level Analysis

1. Null values

Null values in Athletes table

```
SELECT COUNT(*) - COUNT(id) AS id,  
       COUNT(*) - COUNT(age) AS age,  
       COUNT(*) - COUNT(height) AS height,  
       COUNT(*) - COUNT(weight) AS weight  
FROM athletes;
```

We will deal with missing weights when we analyze athlete performance by BMI. Rest is good to go

id bigint	age bigint	height bigint	weight bigint
0	0	0	151

2. Duplicate records in the table

Duplicate values in Athletes table

```
SELECT id,  
       COUNT(*)  
FROM athletes  
GROUP BY id  
ORDER BY COUNT DESC
```

	id integer	count bigint
1	127594	2
2	16592	1
3	84925	1
4	35592	1
5	273	1
6	119817	1

Upon checking it was found it was a duplicate record with 2 ages. So dropping one

Dropping duplicate values

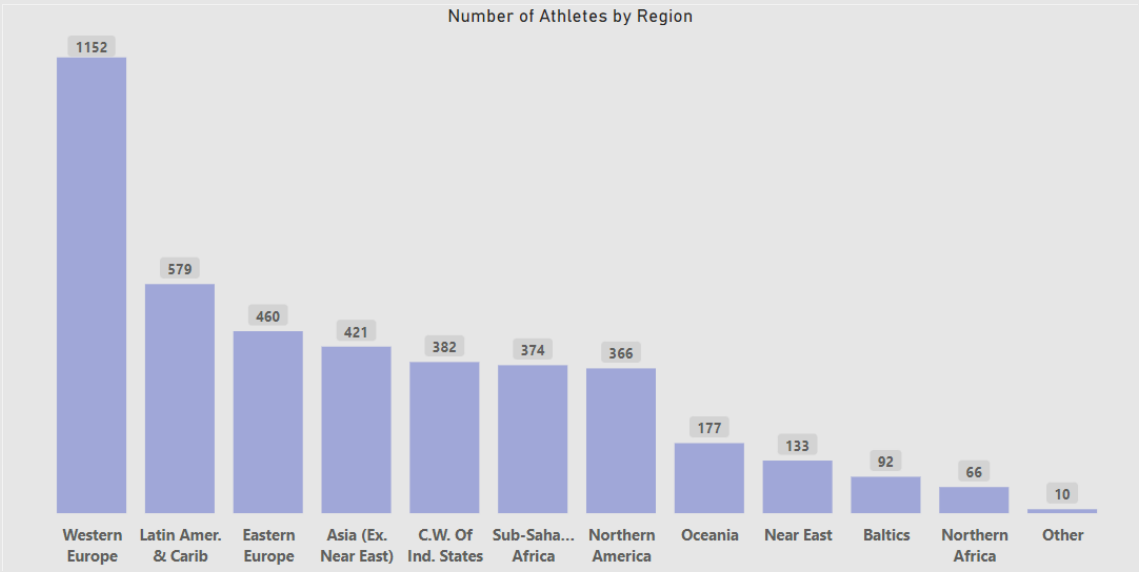
```
DELETE  
FROM athletes  
WHERE id = 127594  
      AND age = 30;
```

3. Athlete Count by Region

Athlete count by region

```
SELECT initcap(coalesce(c.region, 'Other'))
      AS "Region",
      count(DISTINCT a.id)
      AS "Number of Athletes"
FROM athletes a
INNER JOIN games
ON a.id = games.athlete_id
INNER JOIN country c
ON games.country_id = c.id
GROUP BY region
ORDER BY "Number of Athletes" DESC;
```

	Region text	Number of Athletes bigint
1	Western Europe	1152
2	Latin Amer. & Carib	579
3	Eastern Europe	460
4	Asia (Ex. Near East)	421
5	C.W. Of Ind. States	382
6	Sub-Saharan Africa	374
7	Northern America	366
8	Oceania	177
9	Near East	133
10	Baltics	92
11	Northern Africa	66
12	Other	10

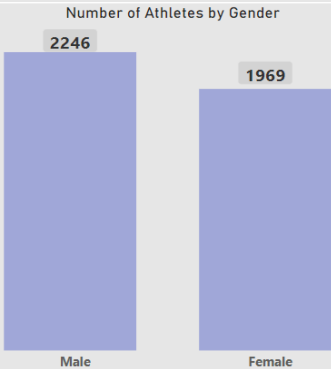


4. Count by Gender

Gender Distribution

```
SELECT gender AS "Gender",
      COUNT(id) AS "Num"
FROM athletes
GROUP BY gender;
```

	Gender character var	Num bigint
1	M	2246
2	F	1969



5. Medal count by Gender

```
Medal count by gender

SELECT gender,
       COUNT(gold) AS gold,
       COUNT(silver) AS silver,
       COUNT(bronze) AS bronze,
       COUNT(gold)+count(silver)+count(bronze)
       AS total
FROM games g
INNER JOIN athletes a
ON g.athlete_id = a.id
GROUP BY gender;
```

	gender character varying (8) 🔒	gold bigint 🔒	silver bigint 🔒	bronze bigint 🔒	total bigint 🔒
1	M	105	97	95	297
2	F	101	97	93	291

Quite similar
distribution of
medals is
observed

6. Medal count by different age groups

We will create age groups with 5 year intervals to see which age group performed best.

```
Create age bins

CREATE TEMP TABLE age_bins AS
(SELECT generate_series(10, 50, 5) AS min_age,
       generate_series(15, 55, 5) AS max_age);

SELECT * FROM age_bins;
```

	min_age integer 🔒	max_age integer 🔒
1	10	15
2	15	20
3	20	25
4	25	30
5	30	35
6	35	40
7	40	45
8	45	50
9	50	55

We will then join and group by these bins to get the breakdown of medals.

```
Medals by age groups

SELECT concat(min_age, ' - ', max_age, ' years')
       AS age,
       COUNT(gold) AS gold,
       COUNT(silver) AS silver,
       COUNT(bronze) AS bronze,
       COUNT(gold) + count(silver) +count(bronze)
       AS total
FROM games g
INNER JOIN athletes a ON g.athlete_id = a.id
INNER JOIN age_bins ON a.age >= min_age
AND a.age < max_age
GROUP BY min_age,max_age
ORDER BY min_age;
```

	age text 🔒	gold bigint 🔒	silver bigint 🔒	bronze bigint 🔒	total bigint 🔒
1	10 - 15 years	0	0	0	0
2	15 - 20 years	22	14	23	59
3	20 - 25 years	74	72	69	215
4	25 - 30 years	68	73	61	202
5	30 - 35 years	38	29	29	96
6	35 - 40 years	3	5	6	14
7	40 - 45 years	1	1	0	2
8	45 - 50 years	0	0	0	0

Winners are largely
concentrated in the 20-30 age
group

6. Medal count by BMI (Body mass index) of Athletes.

Earlier we observed some weight values were missing. So now we will impute them by average weight values grouped across 5 year age bins and 10 cm height bins and gender.
First creating the weight matrix

● ● ● Weight Matrix

```
--create height bins
CREATE TEMP TABLE height_bins AS
(SELECT generate_series(130, 200, 10) min_height,
generate_series(140, 210, 10) AS max_height);

--create a average weight matrix according to
gender,age and height
CREATE TEMP TABLE weight_matrix AS
(SELECT gender,
        min_age,
        max_age,
        min_height,
        max_height,
        round(avg(weight), 2) avg_weight
FROM athletes
JOIN age_bins
ON (athletes.age >= age_bins.min_age AND
    athletes.age < age_bins.max_age)
JOIN height_bins
    ON (athletes.height >= height_bins.min_height
    AND athletes.height < height_bins.max_height)
GROUP BY
    gender,min_age,max_age,min_height,max_height
ORDER BY
    min_age,max_age,min_height,max_height);
```

	gender character varying (8) 🔒	min_age integer 🔒	max_age integer 🔒	min_height integer 🔒	max_height integer 🔒	avg_weight numeric 🔒
1	F	10	15	150	160	45
2	F	10	15	160	170	55.25
3	M	10	15	160	170	52
4	F	10	15	170	180	55
5	F	10	15	180	190	60
6	F	15	20	130	140	37
7	F	15	20	140	150	42.29
8	M	15	20	150	160	71.5
9	F	15	20	150	160	49.95

Creating table with imputed weight

```
Imputed weight

CREATE TABLE athletes_imputed AS
(SELECT id,
        name,
        a.gender,
        age,
        height,
        round(coalesce(weight,
avg_weight)) AS weight
FROM athletes a
JOIN weight_matrix b ON a.gender = b.gender
        AND age >= min_age
        AND age < max_age
        AND height >= min_height
        AND height < max_height)
```

Checking the average weight and null values across the tables

```
Checking

SELECT AVG(weight) AS avg_weight,
        COUNT(*) - COUNT(weight) AS missing_value
FROM athletes
UNION
SELECT AVG(weight) AS avg_weight,
        COUNT(*) - COUNT(weight) AS missing_value
FROM athletes_imputed
```

	avg_weight numeric	missing_value bigint
1	68.10631229235881	0
2	68.37475393700787	151

Which shows us that our imputation was successful

Calculating BMI and distribution of medals according to BMI

```
Medal count by BMI

--creating a temporary BMI table
CREATE TEMP TABLE bmi AS
(WITH cte AS
(SELECT id,round(weight * 10000 / (height * height), 2)AS bmi
FROM athletes_imputed)

SELECT id,bmi,
CASE WHEN bmi < 18.50 THEN 'Underweight'
      WHEN bmi BETWEEN 18.50 AND 24.9 THEN 'Healthy'
      ELSE 'Overweight' END AS category
FROM cte)

--count of medals by BMI
SELECT category,
        count(gold) AS gold,
        count(silver) AS silver,
        count(bronze) AS bronze,
        count(gold) + count(silver) + count(bronze) AS total
FROM games g
INNER JOIN bmi a ON g.athlete_id = a.id
GROUP BY category
```

category text	gold bigint	silver bigint	bronze bigint	total bigint
Overweight	21	16	19	56
Healthy	178	165	155	498
Underweight	7	13	14	34

Not surprisingly most winners are from healthy category however it is interesting to note that there are some winners who have overweight under normal considerations of BMI.

7. Best performing Athletes

Best Performing Athletes

```
SELECT a.name,
       a.gender,
       initcap(substr(REPLACE(c.name, '.', ''),
                       POSITION('-' in REPLACE(c.name, '.', '')) + 2))
AS "Country",
       COUNT(gold) AS gold,
       COUNT(silver) AS silver,
       COUNT(bronze) AS bronze,
       COUNT(gold)+COUNT(silver)+COUNT(bronze)
AS total
FROM athletes a
INNER JOIN games g ON a.id = g.athlete_id
INNER JOIN country c ON g.country_id = c.id
GROUP BY a.name,
         a.gender,
         c.name
ORDER BY total DESC
LIMIT 15;
```

	name character varying (255)	gender character var	Country text	gold bigint	silver bigint	bronze bigint	total bigint
1	Michael Fred Phelps, II	M	United States	5	1	0	6
2	Kathleen Genevieve "...	F	United States	4	1	0	5
3	Simone Arianne Biles	F	United States	4	0	1	5
4	Simone Ashley Manuel	F	United States	2	2	0	4
5	Penelope "Penny" Ole...	F	Canada	1	1	2	4
6	Nathan Ghar-Jun Adri...	M	United States	2	0	2	4
7	Katinka Hossz	F	Hungary	3	1	0	4
8	Emma McKeon	F	Australia	1	2	1	4
9	Madeline Jane "Maya..."	F	United States	2	1	1	4
10	Andre De Grasse	M	Canada	0	1	2	3
11	Kyle Chalmers	M	Australia	1	0	2	3
12	Alexandra Rose "Aly" ...	F	United States	1	2	0	3
13	Elaine Thompson	F	Jamaica	2	1	0	3
14	Marina Charlotte Kalla	F	Sweden	1	2	0	3
15	Allyson Michelle Felix	F	United States	2	1	0	3

Games Level Analysis

Time of games

```
-- time duration of games

SELECT type,
       max(date_part('year', YEAR))
FROM games
GROUP BY type;
```

type	max
text	double precis
winter_games	2014
summer_games	2016

Sports by game type

```
--sport by type

SELECT distinct(type),
       sport
FROM games
ORDER BY type;
```

type	sport
text	character varying (255)
summer_ga...	Gymnastics
summer_ga...	Swimming
summer_ga...	Track and Field
winter_games	Alpine Skiing
winter_games	Biathlon
winter_games	Cross Country Skiing

```
--number of events by type, country and athlete participation and total medal counts

SELECT TYPE,
       count(DISTINCT event) AS EVENTS,
       count(DISTINCT country_id) AS countries_participated,
       count(DISTINCT athlete_id) AS athletes_participated,
       count(gold) + count(silver) + count(bronze) AS total_medals
FROM games
GROUP BY TYPE;
```

	type	events	countries_participated	athletes_participated	total_medals
	text	bigint	bigint	bigint	bigint
1	summer_ga...	95	203	3407	449
2	winter_games	32	78	806	139

Number of events conducted by gender

```
--number of events by gender category

with cte as
(SELECT DISTINCT event,
 CASE
  WHEN lower(event) like '%women%' THEN 'Womens Events'
  WHEN lower(event) like '%men%' THEN 'Mens Events'
  ELSE 'Other' END AS category
 FROM games )

SELECT category,count(event) AS COUNT
FROM cte
GROUP BY category
```

category text	count bigint
Womens Events	62
Mens Events	65

Almost equal
number of events
are conducted
across genders

Medalists by Event Name

```
--all event with winner names

SELECT g.event,
       max(CASE WHEN gold = 1 THEN a.name ELSE NULL END) AS "Gold Medallist",
       max(CASE WHEN silver = 1 THEN a.name ELSE NULL END) AS "Silver Medallist",
       max(CASE WHEN bronze = 1 THEN a.name ELSE NULL END) "Bronze Medallist"
FROM games g
INNER JOIN athletes a ON g.athlete_id = a.id
GROUP BY g.event
ORDER BY g.event;

--we can use string_agg to list all winners however for simplicity we're using max to output 1
player
```

	event character varying (255)	Gold Medallist text	Silver Medallist text	Bronze Medallist text
1	Alpine Skiing Men's Combined	Sandro Viletta	Ivica Kosteli	Christof Innerhofer
2	Alpine Skiing Men's Downhill	Matthias Mayer	Christof Innerhofer	Kjetil Jansrud
3	Alpine Skiing Men's Giant Slalom	Theodore Sharp "Ted" Ligety	Steve Missillier	Alexis Pinturault
4	Alpine Skiing Men's Slalom	Mario Matt	Marcel Hirscher	Henrik Kristoffersen
5	Alpine Skiing Men's Super G	Kjetil Jansrud	Andrew Weibrecht	Samuel Bode Miller
6	Alpine Skiing Women's Combined	Maria Hfl-Riesch	Nicole Hosp	Julia Marie Mancuso (-Fish)
7	Alpine Skiing Women's Downhill	Klementina "Tina" Maze	[null]	Lara Gut
8	Alpine Skiing Women's Giant Slalom	Klementina "Tina" Maze	Anna Fenninger (-Veith)	Viktoria Rebensburg
9	Alpine Skiing Women's Slalom	Mikaela Pauline Shiffrin	Marlies Schild (-Raich)	Kathrin Zettel
10	Alpine Skiing Women's Super G	Anna Fenninger (-Veith)	Maria Hfl-Riesch	Nicole Hosp
11	Biathlon Men's 10 kilometres Sprint	Ole Einar Bjrndalen	Dominik Landertinger	Jaroslav Soukup
12	Biathlon Men's 12.5 kilometres Pursuit	Martin Fourcade	Ondej Moravec	Jean-Guillaume Batrix
13	Biathlon Men's 15 kilometres Mass Start	Emil Hegle Svendsen	Martin Fourcade	Ondej Moravec
14	Biathlon Men's 20 kilometres	Martin Fourcade	Erik Lesser	Yevgeny Aleksandrovich Garanichev
15	Biathlon Men's 4 x 7.5 kilometres Relay	Yevgeny Romanovich Ustyugov	Simon Schempp	Simon Eder

Country Level Analysis

Athlete count by country

```
--athlete count by country top 15

SELECT c.name as Country,
       count(DISTINCT a.id) AS "Number of Athletes"
FROM athletes a
INNER JOIN games
ON a.id = games.athlete_id
INNER JOIN country c
ON games.country_id = c.id
GROUP BY c.name
ORDER BY "Number of Athletes" DESC;
```

country character varying (255)	Number of At bigint
U.S.A. - United States	228
ger - Germany	156
CAN - Canada	133
fra - france	128
ITA - Italy	123
G.BR - Great Britain	119

We observe inconsistent formatting in country column so we need to clean that column

Cleaned country table

```
CREATE TABLE country_clean AS
(SELECT
id,
LEFT(UPPER(TRIM(REPLACE(name, '.', ''))), 3) AS country_code,
initcap(substr(TRIM(REPLACE(name, '.', '')),POSITION('-', TRIM(REPLACE(name, '.', '')))+2))
AS name,
gdp,
pop_in_million,
initcap(coalesce(region, 'Other')) AS region
FROM country);
```

Athlete count by country cleaned

```
--athlete count by country top 15

SELECT c.name AS country,
       COUNT(DISTINCT a.id) AS "Number of Athletes"
FROM athletes a
INNER JOIN games
ON a.id = games.athlete_id
INNER JOIN country_clean c
ON games.country_id = c.id
GROUP BY c.name
ORDER BY "Number of Athletes" DESC
LIMIT 15;
```

	country text	Number of Athletes bigint
1	United States	228
2	Germany	156
3	Canada	133
4	France	128
5	Italy	123
6	Great Britain	119
7	China	116
8	Brazil	112
9	Japan	110
10	Australia	110
11	Poland	110
12	Ukraine	97
13	Russia	89
14	Spain	84
15	Switzerland	70

Top 10 countries by medal count

Top Countries by Medal Count

```
--medals by country top 10
```

```
SELECT c.name,  
       COUNT(gold) AS gold,  
       COUNT(silver) AS silver,  
       COUNT(bronze) AS bronze,  
       COUNT(gold) + COUNT(silver) +  
       COUNT(bronze) AS total  
FROM country_clean c  
INNER JOIN games g  
ON c.id = g.country_id  
GROUP BY c.name  
ORDER BY total DESC  
LIMIT 10;
```

	name text	gold bigint	silver bigint	bronze bigint	total bigint
1	United States	85	31	22	138
2	Russia	6	26	7	39
3	Australia	7	15	12	34
4	Great Britain	5	14	14	33
5	Jamaica	11	17	2	30
6	Canada	2	2	22	26
7	France	2	12	10	24
8	Norway	9	3	11	23
9	Japan	8	6	8	22
10	China	3	4	15	22

Medals distribution by GDP and Population Categories

We will use percentile function to calculate bins for GDP and Population and categorize the countries.

Top Countries by Medal Count

```
--we will create bins using percentile function for gdp and population
```

```
SELECT percentile_disc(0.20) within group (order by gdp)  
, percentile_disc(0.40) within group (order by gdp)  
, percentile_disc(0.60) within group (order by gdp)  
, percentile_disc(0.80) within group (order by gdp)  
, percentile_disc(1) within group (order by gdp)  
FROM country_clean;
```

```
--we will similarly calculate for population
```

```
--we will categorize into bins
```

```
create temp table country_category as (  
SELECT id,  
       name,  
       CASE  
         WHEN gdp <= 2921136786 THEN 'Very Low Income'  
         WHEN gdp <= 12344677303 THEN 'Low Income'  
         WHEN gdp <= 45038518439 THEN 'Moderate Income'  
         WHEN gdp <= 256142857143 THEN 'High Income'  
         ELSE 'Very High Income' END AS Income_Category,  
       CASE  
         WHEN pop_in_million <= 0.543935 THEN 'Very Low Population'  
         WHEN pop_in_million <= 4.284332 THEN 'Low Population'  
         WHEN pop_in_million <= 10.428074 THEN 'Moderate Population'  
         WHEN pop_in_million <= 30.309135 THEN 'High Population'  
         ELSE 'Very High Population' END AS Population_Category  
FROM country_clean);
```


Medal count by Population

```
Medal count by Countries categorised by population

--medal count by population

SELECT
    population_category,
    COUNT(DISTINCT id) AS country_count,
    COUNT(gold) AS gold,
    COUNT(silver) AS silver,
    COUNT(bronze) AS bronze,
    COUNT(gold)+COUNT(silver)+COUNT(bronze)
    AS total
FROM country_category c
INNER JOIN games g
ON c.id = g.country_id
GROUP BY population_category
ORDER BY total DESC;
```

	population_category text	country_count bigint	gold bigint	silver bigint	bronze bigint	total bigint
1	Very High Population	40	136	124	121	381
2	Moderate Population	40	38	28	39	105
3	High Population	41	15	21	17	53
4	Low Population	41	16	20	6	42
5	Very Low Population	41	1	1	5	7

Medal count by GDP

```
Medal count by Countries categorised by GDP

--medal count by GDP
SELECT income_category,
    COUNT(DISTINCT id) AS country_count,
    COUNT(gold) AS gold,
    COUNT(silver) AS silver,
    COUNT(bronze) AS bronze,
    COUNT(gold) + COUNT(silver) + COUNT(bronze) AS total
FROM country_category c
INNER JOIN games g ON c.id = g.country_id
GROUP BY income_category
ORDER BY total DESC;
```

	income_category text	country_count bigint	gold bigint	silver bigint	bronze bigint	total bigint
1	Very High Income	40	162	147	155	464
2	High Income	41	29	27	24	80
3	Moderate Income	40	12	18	4	34
4	Low Income	41	2	0	5	7
5	Very Low Income	41	1	2	0	3

Medal count by GDP per capita







```
Medal count by Countries categorised by GDP per capita

--medal count by gdp per capita
--calculating gdp per capita
WITH cte AS
(SELECT id,
      name,
      gdp / (pop_in_million * 1000000.0) AS gdp_per_capita
FROM country_clean
WHERE pop_in_million <> 0 ),

--classified according to world bank data

cte2 AS
(SELECT id,
      name,
      CASE
      WHEN gdp_per_capita <= 1035 THEN 'low income'
      WHEN gdp_per_capita BETWEEN 1036 AND 4085 THEN 'lower middle income'
      WHEN gdp_per_capita BETWEEN 4086 AND 12615 THEN 'upper middle income'
      ELSE 'high income' END AS country_category
FROM cte)

SELECT country_category,
      count(DISTINCT id) AS country_count,
      count(gold) AS gold,
      count(silver) AS silver,
      count(bronze) AS bronze,
      count(gold) + count(silver) + count(bronze) AS total
FROM cte2
INNER JOIN games ON cte2.id = games.country_id
GROUP BY country_category;
```

	country_category  text	country_count  bigint	gold  bigint	silver  bigint	bronze  bigint	total  bigint
1	high income	62	170	149	155	474
2	low income	37	3	3	5	11
3	lower middle income	51	11	7	3	21
4	upper middle income	48	22	35	25	82