

Title: Tokyo Olympics 2020: Unveiling the Data Behind the Games

1. Database implementation

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to db-project-379821.
Use "gcloud config set project [PROJECT ID]" to change to a different project.
amruthaakkalam00@cloudshell:~ (db-project-379821)$ gcloud sql connect db-project-team017 --user=root --quiet
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1831
Server version: 8.0.26-google (Google)
```

```
Copyright (c) 2000, 2023, Oracle and/or its affiliates.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> show databases;
```

```

+-----+
| Database |
+-----+
| Project-017 |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
```

```
mysql> show databases;
```

```

+-----+
| Database |
+-----+
| Project-017 |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)
```

```
mysql> show tables;
```

```

+-----+
| Tables_in_Project-017 |
+-----+
| Athletes |
| Coaches |
| Gender |
| Medals |
| Teams |
| Top3 |
+-----+
6 rows in set (0.00 sec)
```

2. DDL commands

```
Cloud Shell
Terminal (db-project-379821) x + - Open editor

mysql> CREATE TABLE IF NOT EXISTS Teams(
  -> Team_Name VARCHAR(50) NOT NULL,
  -> Discipline VARCHAR(50) NOT NULL,
  -> NOC VARCHAR(50) NOT NULL,
  -> T_Event VARCHAR(50) NOT NULL,
  -> PRIMARY KEY (NOC, Discipline, T_Event)
  -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE INDEX fk_Top3 on TEAMS (Discipline, T_Event);
ERROR 1146 (4202): Table 'Project-017.TEAMS' doesn't exist
mysql> CREATE INDEX fk_Top3 on Teams (Discipline, T_Event);
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> CREATE TABLE IF NOT EXISTS Top3(
  -> Discipline VARCHAR(50) NOT NULL,
  -> D_Event VARCHAR(50) NOT NULL,
  -> GOLD VARCHAR(50) NOT NULL,
  -> SILVER VARCHAR(50) NOT NULL,
  -> BRONZE VARCHAR(50) NOT NULL,
  -> PRIMARY KEY (Discipline, D_Event),
  -> FOREIGN KEY (Discipline, D_Event) REFERENCES Teams(Discipline, T_Event)
  -> ON DELETE CASCADE
  -> ON UPDATE CASCADE
  -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> CREATE TABLE IF NOT EXISTS Gender(
  -> Discipline VARCHAR(50) NOT NULL,
  -> Females INT NOT NULL,
  -> Males INT NOT NULL,
  -> Total INT NOT NULL,
  -> PRIMARY KEY (Discipline),
  -> );
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ')' at line 7
mysql> CREATE TABLE IF NOT EXISTS Gender( Discipline VARCHAR(50) NOT NULL, Females INT NOT NULL, Males INT NOT NULL, Total INT NOT NULL, PRIMARY KEY (Discipline) );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> CREATE TABLE IF NOT EXISTS Medals(
  -> G_Rank INT NOT NULL,
  -> NOC VARCHAR(50) NOT NULL,
  -> GOLD INT NOT NULL,
  -> SILVER INT NOT NULL,
  -> BRONZE INT NOT NULL,
  -> Total INT NOT NULL,
  -> Total_Rank INT NOT NULL,
  -> PRIMARY KEY (NOC)
  -> );
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql>
```

```
mysql> CREATE TABLE IF NOT EXISTS Athletes( A_Name VARCHAR(50) NOT NULL, NOC VARCHAR(50) NOT NULL, Discipline VARCHAR(50) NOT NULL, PRIMARY KEY (A_Name, NOC, Discipline) );
Query OK, 0 rows affected (0.04 sec)
```

3. Table Insertions

```
mysql> select count(*) from Athletes;
+-----+
| count(*) |
+-----+
| 11084 |
+-----+
1 row in set (0.01 sec)

mysql> Select count(*) from Teams;
+-----+
| count(*) |
+-----+
| 5417 |
+-----+
1 row in set (0.00 sec)
```

4. Advanced Queries:

1. Find all the American Athletes in the Discipline with the highest number of events.

```
SELECT A.A_Name, A.NOC, A.Discipline
FROM Athletes A
WHERE A.discipline = (
    SELECT Discipline
    FROM Top3
    GROUP BY Discipline
    HAVING COUNT(Discipline) >= ALL(
        SELECT COUNT(t.Discipline) FROM Top3 t GROUP BY t.Discipline
    )
)
AND A.NOC LIKE 'United States of America'
ORDER BY A.A_Name
LIMIT 15;
```

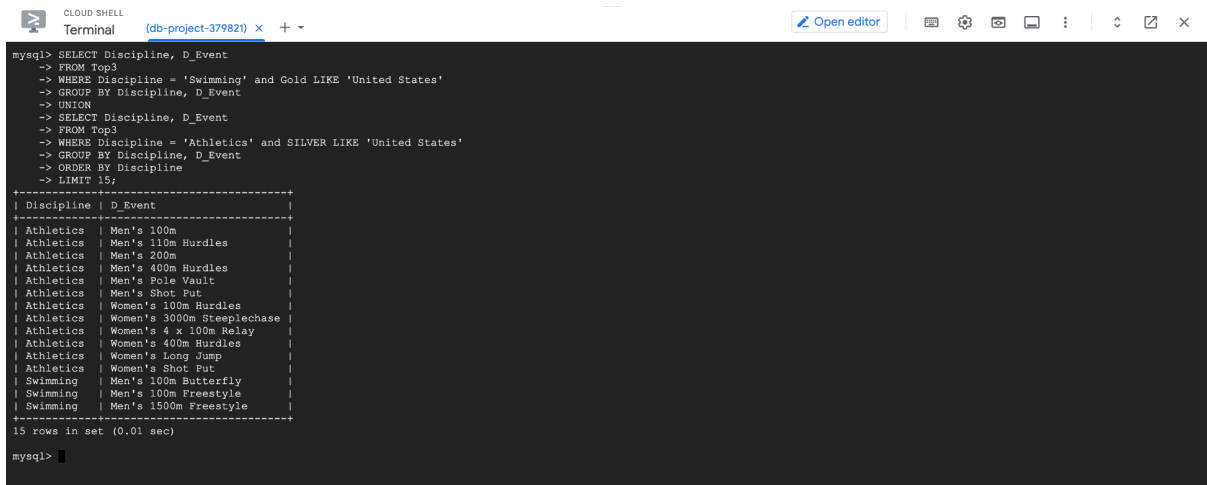
```
CLOUD SHELL
Terminal (db-project-379820) X + -
Open editor

mysql> SELECT A.A_Name, A.NOC, A.Discipline
-> FROM Athletes A
-> WHERE A.discipline = (
-> SELECT Discipline
-> FROM Top3
-> GROUP BY Discipline
-> HAVING COUNT(Discipline) >= ALL(
-> SELECT COUNT(t.Discipline) FROM Top3 t GROUP BY t.Discipline
-> )
-> )
-> AND A.NOC LIKE 'United States of America'
-> ORDER BY A.A_Name
-> LIMIT 15;
+-----+-----+-----+
| A_Name | NOC | Discipline |
+-----+-----+-----+
| ABDIRAHMAN Abdi | United States of America | Athletics |
| ALLEN Devon | United States of America | Athletics |
| ALLMAN Valarie | United States of America | Athletics |
| ANDERSEN Brooke | United States of America | Athletics |
| ANDERSON Shae | United States of America | Athletics |
| AQUILLA Adelaide | United States of America | Athletics |
| ASHLEY Whitney | United States of America | Athletics |
| BAILEY JR Aldrich | United States of America | Athletics |
| BAKER Ronnie | United States of America | Athletics |
| BASTIEN Steven | United States of America | Athletics |
| BATTLE Anavia | United States of America | Athletics |
| BEDNAREK Kenneth | United States of America | Athletics |
| BENARD Chris | United States of America | Athletics |
| BENJAMIN Rai | United States of America | Athletics |
| BERRY Gwen | United States of America | Athletics |
+-----+-----+-----+
15 rows in set (0.00 sec)

mysql>
```

2. List all events in Athletics and Swimming in which the USA secured a Silver medal in Athletics and Gold in Swimming

```
SELECT Discipline, D_Event
FROM Top3
WHERE Discipline = 'Swimming' and Gold LIKE 'United States'
GROUP BY Discipline, D_Event
UNION
SELECT Discipline, D_Event
FROM Top3
WHERE Discipline = 'Athletics' and SILVER LIKE 'United States'
GROUP BY Discipline, D_Event
ORDER BY Discipline
LIMIT 15;
```



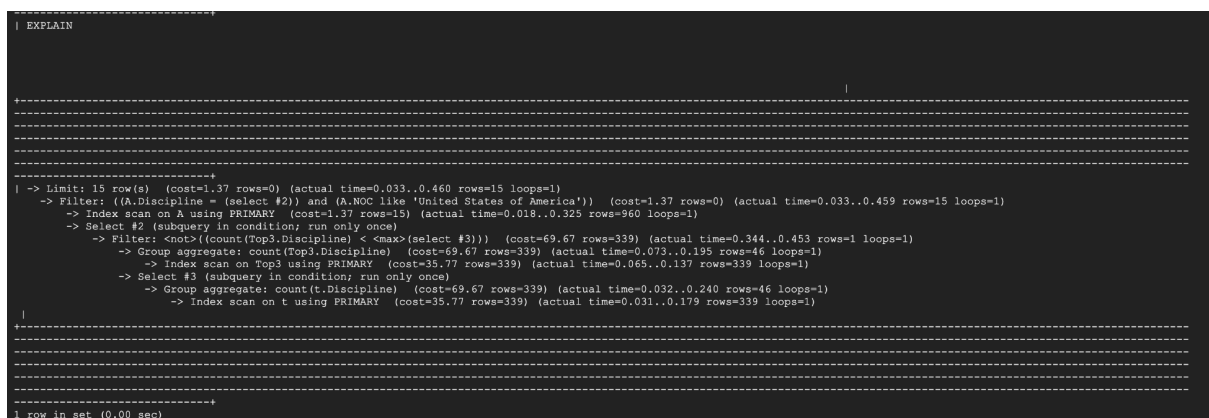
```
mysql> SELECT Discipline, D_Event
-> FROM Top3
-> WHERE Discipline = 'Swimming' and Gold LIKE 'United States'
-> GROUP BY Discipline, D_Event
-> UNION
-> SELECT Discipline, D_Event
-> FROM Top3
-> WHERE Discipline = 'Athletics' and SILVER LIKE 'United States'
-> GROUP BY Discipline, D_Event
-> ORDER BY Discipline
-> LIMIT 15;
+-----+-----+
| Discipline | D_Event |
+-----+-----+
| Athletics | Men's 100m |
| Athletics | Men's 110m Hurdles |
| Athletics | Men's 200m |
| Athletics | Men's 400m Hurdles |
| Athletics | Men's Pole Vault |
| Athletics | Men's Shot Put |
| Athletics | Women's 100m Hurdles |
| Athletics | Women's 3000m Steeplechase |
| Athletics | Women's 4 x 100m Relay |
| Athletics | Women's 400m Hurdles |
| Athletics | Women's Long Jump |
| Athletics | Women's Shot Put |
| Swimming | Men's 100m Butterfly |
| Swimming | Men's 100m Freestyle |
| Swimming | Men's 1500m Freestyle |
+-----+-----+
15 rows in set (0.01 sec)

mysql>
```

5. Indexing

Query1:

Before Indexing:



```
| EXPLAIN
+-----+-----+
| |
+-----+-----+
| -> Limit: 15 row(s) (cost=1.37 rows=0) (actual time=0.033..0.460 rows=15 loops=1)
|   -> Filter: ((A.Discipline = (select #2)) and (A.NOC like 'United States of America')) (cost=1.37 rows=0) (actual time=0.033..0.459 rows=15 loops=1)
|     -> Index scan on A using PRIMARY (cost=1.37 rows=15) (actual time=0.018..0.325 rows=960 loops=1)
|     -> Select #2 (subquery in condition; run only once)
|       -> Filter: <not>(count(Top3.Discipline) < <max>(select #3))) (cost=69.67 rows=339) (actual time=0.344..0.453 rows=1 loops=1)
|         -> Group aggregate: count(Top3.Discipline) (cost=69.67 rows=339) (actual time=0.073..0.195 rows=46 loops=1)
|           -> Index scan on Top3 using PRIMARY (cost=35.77 rows=339) (actual time=0.065..0.137 rows=339 loops=1)
|           -> Select #3 (subquery in condition; run only once)
|             -> Group aggregate: count(t.Discipline) (cost=69.67 rows=339) (actual time=0.032..0.240 rows=46 loops=1)
|               -> Index scan on t using PRIMARY (cost=35.77 rows=339) (actual time=0.031..0.179 rows=339 loops=1)
|
+-----+-----+
1 row in set (0.00 sec)
```

Index1: CREATE INDEX idx_country ON Athletes(NOC);

[illegible]

```
Index2: CREATE INDEX idx_a_name ON Athletes(A_Name);
```

[illegible]

```
Index3: CREATE INDEX idx_discipline_top3 ON Top3(Discipline);
```

[illegible]

Analysis:

The best speed up for lower bound and upper bound is achieved by Index 3 because the primary operation involves aggregation on Discipline in Top3.

Adding indexes has definitely made the operations faster than before indexing as we target the columns used in the query to index.

Creating an index on A name actually increases the time considerably

Query2:

Before Indexing:

```

| EXPLAIN
|
|-----+
|
|-----+
| -> Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.020..0.022 rows=15 loops=1)
|   -> Sort: Discipline, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.019..0.020 rows=15 loops=1)
|     -> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.003 rows=23 loops=1)
|       -> Union materialize with deduplication (cost=3.43..5.66 rows=9) (actual time=0.118..0.121 rows=23 loops=1)
|         -> Filter: ((Top3.Gold like 'United States')) (cost=1.04 rows=4) (actual time=0.032..0.043 rows=11 loops=1)
|           -> Index lookup on Top3 using PRIMARY (Discipline='Swimming') (cost=1.04 rows=35) (actual time=0.027..0.034 rows=35 loops=1)
|         -> Filter: (Top3.Silver like 'United States') (cost=1.20 rows=5) (actual time=0.022..0.034 rows=12 loops=1)
|           -> Index lookup on Top3 using PRIMARY (Discipline='Athletics') (cost=1.20 rows=48) (actual time=0.021..0.028 rows=48 loops=1)
|       |
|       |-----+
|       |
|       |-----+
| 1 row in set (0.00 sec)

```

Index1: CREATE INDEX idx_gold ON Top3(Gold);

```

| EXPLAIN
|
|-----+
|
|-----+
| -> Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.030..0.032 rows=15 loops=1)
|   -> Sort: Discipline, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.030..0.031 rows=15 loops=1)
|     -> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.003 rows=23 loops=1)
|       -> Union materialize with deduplication (cost=5.58..7.93 rows=16) (actual time=0.150..0.153 rows=23 loops=1)
|         -> Filter: ((Top3.Discipline = 'Swimming') and (Top3.Gold like 'United States')) (cost=2.60 rows=11) (actual time=0.031..0.040 rows=11 loops=1)
|           -> Index range scan on Top3 using idx_gold (cost=2.60 rows=11) (actual time=0.027..0.032 rows=11 loops=1)
|         -> Filter: (Top3.Discipline = 'Swimming') and (Top3.Gold like 'United States') (cost=2.60 rows=11) (actual time=0.027..0.032 rows=11 loops=1)
|           -> Index lookup on Top3 using PRIMARY (Discipline='Athletics') (cost=1.20 rows=48) (actual time=0.025..0.032 rows=48 loops=1)
|       |
|       |-----+
|       |
|       |-----+
| 1 row in set (0.01 sec)

```

Index2: CREATE INDEX idx_discipline_silver ON Top3(Discipline, Silver);;

```

| EXPLAIN
|
|-----+
|
|-----+
| -> Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.025..0.027 rows=15 loops=1)
|   -> Sort: Discipline, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.025..0.025 rows=15 loops=1)
|     -> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.003 rows=23 loops=1)
|       -> Union materialize with deduplication (cost=7.86..10.26 rows=23) (actual time=0.108..0.111 rows=23 loops=1)
|         -> Filter: ((Top3.Discipline = 'Swimming') and (Top3.Gold like 'United States')) (cost=2.60 rows=11) (actual time=0.033..0.041 rows=11 loops=1)
|           -> Index range scan on Top3 using idx_gold (cost=2.60 rows=11) (actual time=0.028..0.032 rows=11 loops=1)
|         -> Filter: ((Top3.Discipline = 'Athletics') and (Top3.Silver like 'United States')) (cost=2.86 rows=12) (actual time=0.013..0.020 rows=12 loops=1)
|           -> Index range scan on Top3 using idx_discipline_silver (cost=2.86 rows=12) (actual time=0.013..0.015 rows=12 loops=1)
|       |
|       |-----+
|       |
|       |-----+
| 1 row in set (0.00 sec)

```

Index3: CREATE INDEX idx_discipline_top3 ON Top3(Discipline,Gold);

```

| EXPLAIN
|
|-----+
|
|-----+
| -> Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.032..0.034 rows=15 loops=1)
|   -> Sort: Discipline, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.032..0.033 rows=15 loops=1)
|     -> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.003 rows=23 loops=1)
|       -> Union materialize with deduplication (cost=7.86..10.26 rows=23) (actual time=0.114..0.117 rows=23 loops=1)
|         -> Filter: ((Top3.Discipline = 'Swimming') and (Top3.Gold like 'United States')) (cost=2.60 rows=11) (actual time=0.031..0.039 rows=11 loops=1)
|           -> Index range scan on Top3 using idx_gold (cost=2.60 rows=11) (actual time=0.027..0.032 rows=11 loops=1)
|         -> Filter: ((Top3.Discipline = 'Athletics') and (Top3.Silver like 'United States')) (cost=2.86 rows=12) (actual time=0.013..0.019 rows=12 loops=1)
|           -> Index range scan on Top3 using idx_discipline_silver (cost=2.86 rows=12) (actual time=0.012..0.015 rows=12 loops=1)
|       |
|       |-----+
|       |
|       |-----+
| 1 row in set (0.00 sec)

```

Range scans are faster than lookup, hence the last 2 indexes are faster. Moreover having an index just over Gold, doesn't prove useful, because the aggregate attribute is discipline.