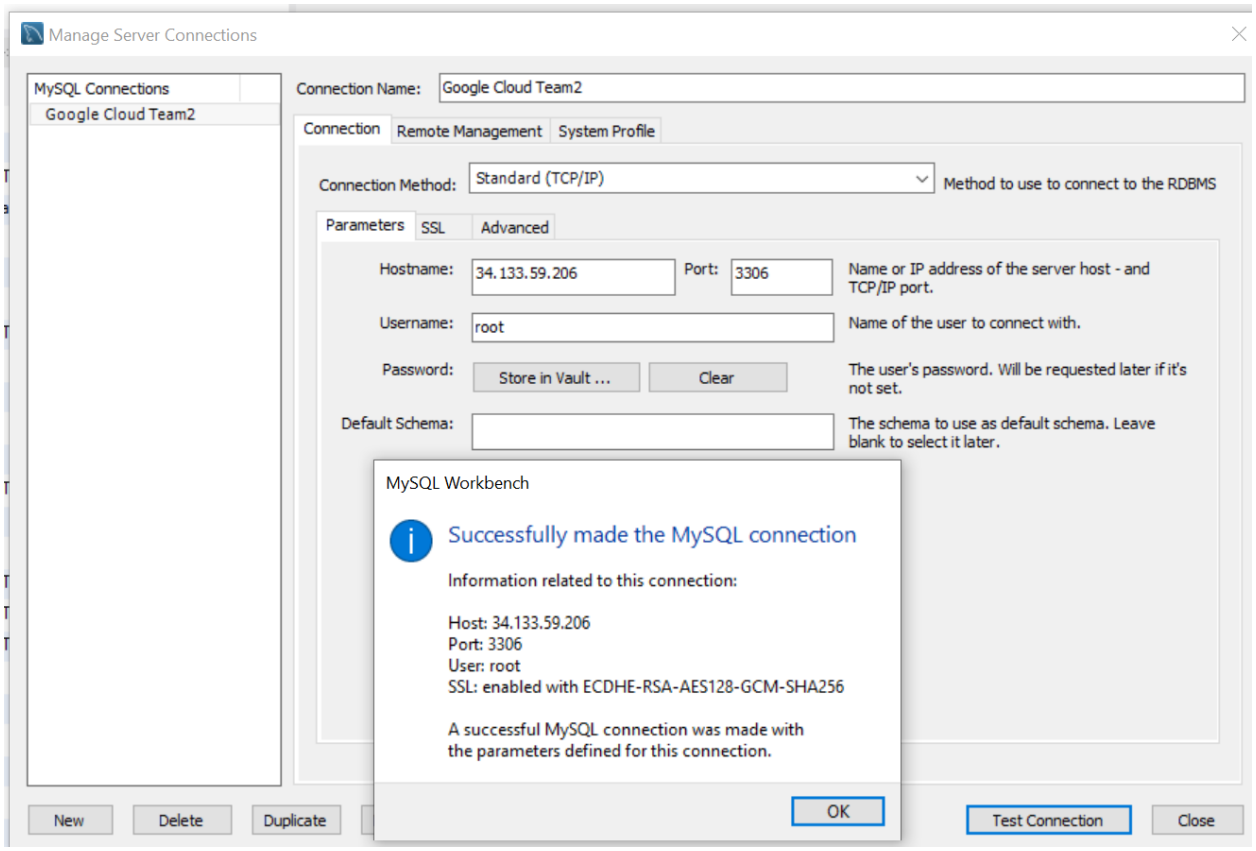
A. Create a markdown or pdf called “Database Design” in the doc folder

GCP Connection:



DDL for Table Creation:

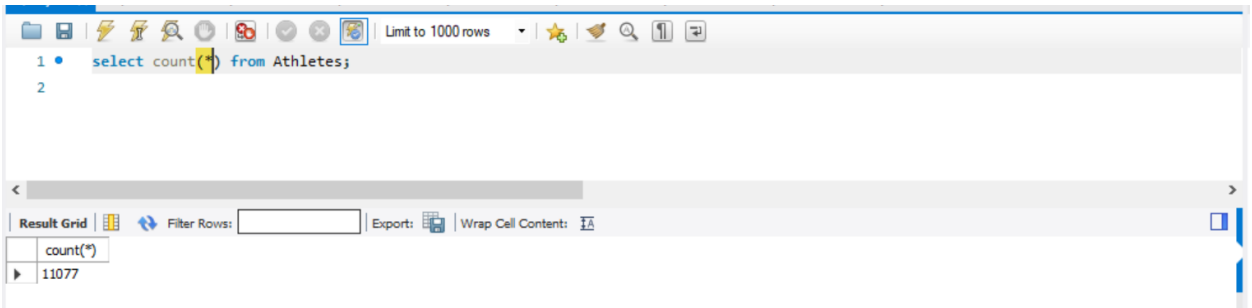
```sql
CREATE TABLE Athletes
(
   Name VARCHAR(512) NOT NULL,
   NOC VARCHAR(512) NOT NULL,
   Discipline  VARCHAR(512) NOT NULL,
   PRIMARY KEY (Name, Discipline)
   FOREIGN KEY (NOC)
);
CREATE TABLE Coaches
(
   Name VARCHAR(512),
   NOC VARCHAR(512),
   Discipline  VARCHAR(512),
   Event VARCHAR(512),
   PRIMARY KEY (Name, Event)
   FOREIGN KEY (NOC)

);
CREATE TABLE Medals
(
   NOC VARCHAR(512),
   RankOverall INT,
   Gold INT,
   Silver INT,
   Bronze INT,
   Total INT,
   RankByNumberOfMedals INT
   PRIMARY KEY (NOC)
   FOREIGN KEY (NOC)
);
CREATE TABLE MedalsByAthlete
(
   athlete_name VARCHAR(512),
   event VARCHAR(512),
   NOC VARCHAR(512),
   discipline  VARCHAR(512),
   medal_type VARCHAR(512),
   medal_date VARCHAR(512),
   PRIMARY KEY (athlete_name, event)
```

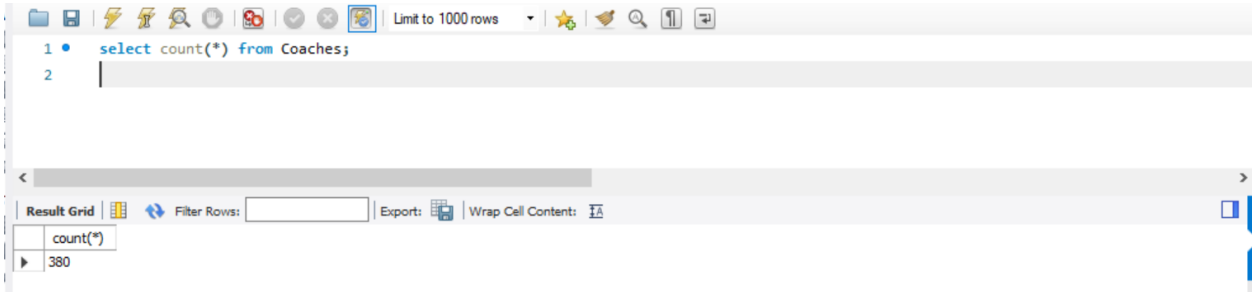
```
);
CREATE TABLE Officials
(
   name VARCHAR(512),
   gender          VARCHAR(512),
   country         VARCHAR(512),
   discipline      VARCHAR(512),
   role   VARCHAR(512),
          //foreign key is missing
   PRIMARY KEY (name, discipline)
   FOREIGN KEY (discipline)
);
```

Number of Entries per table:

Athletes:



Coaches:



Medals:

```
select count(*) from Medals;
```

count(*)
93

MedalsByAthlete:

```
select count(*) from MedalsByAthlete;
```

count(*)
2401

Officials:

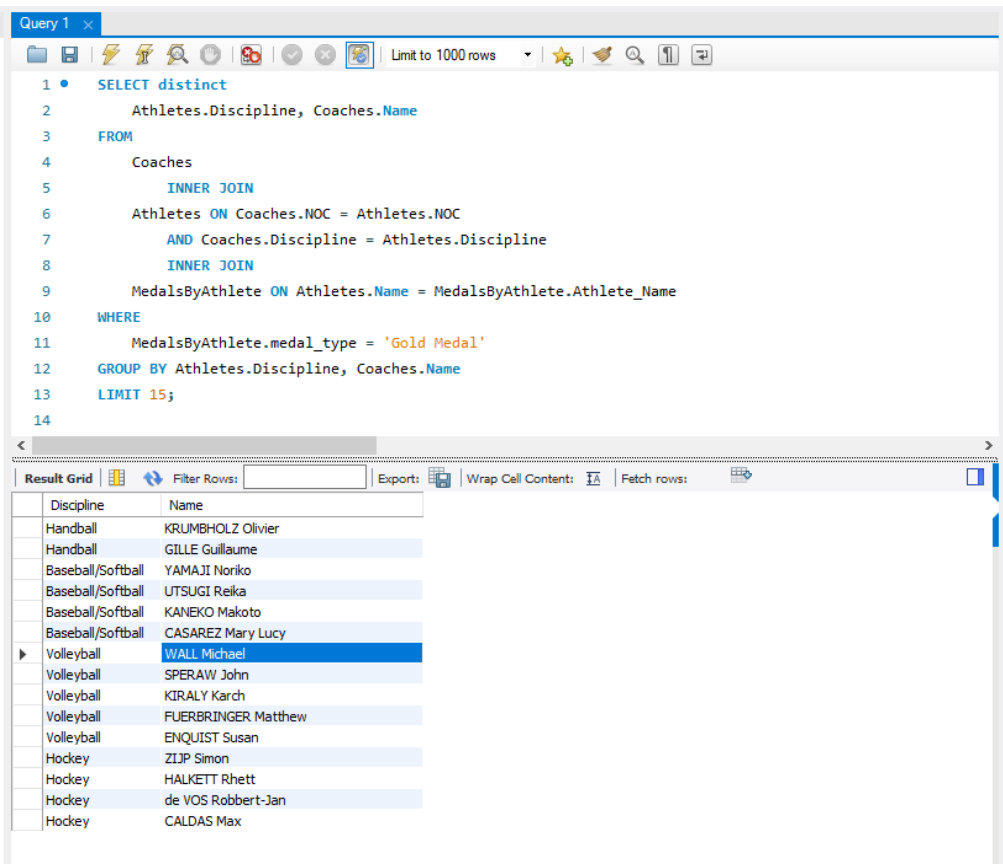
```
select count(*) from Officials;
```

count(*)
1023

#1 Advanced SQL Query

```
SELECT distinct
   Athletes.Discipline, Coaches.Name
FROM
   Coaches
      INNER JOIN
   Athletes ON Coaches.NOC = Athletes.NOC
      AND Coaches.Discipline = Athletes.Discipline
      INNER JOIN
   MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
WHERE
   MedalsByAthlete.medal_type = 'Gold Medal'
GROUP BY Athletes.Discipline, Coaches.Name
LIMIT 15;
```

Image:



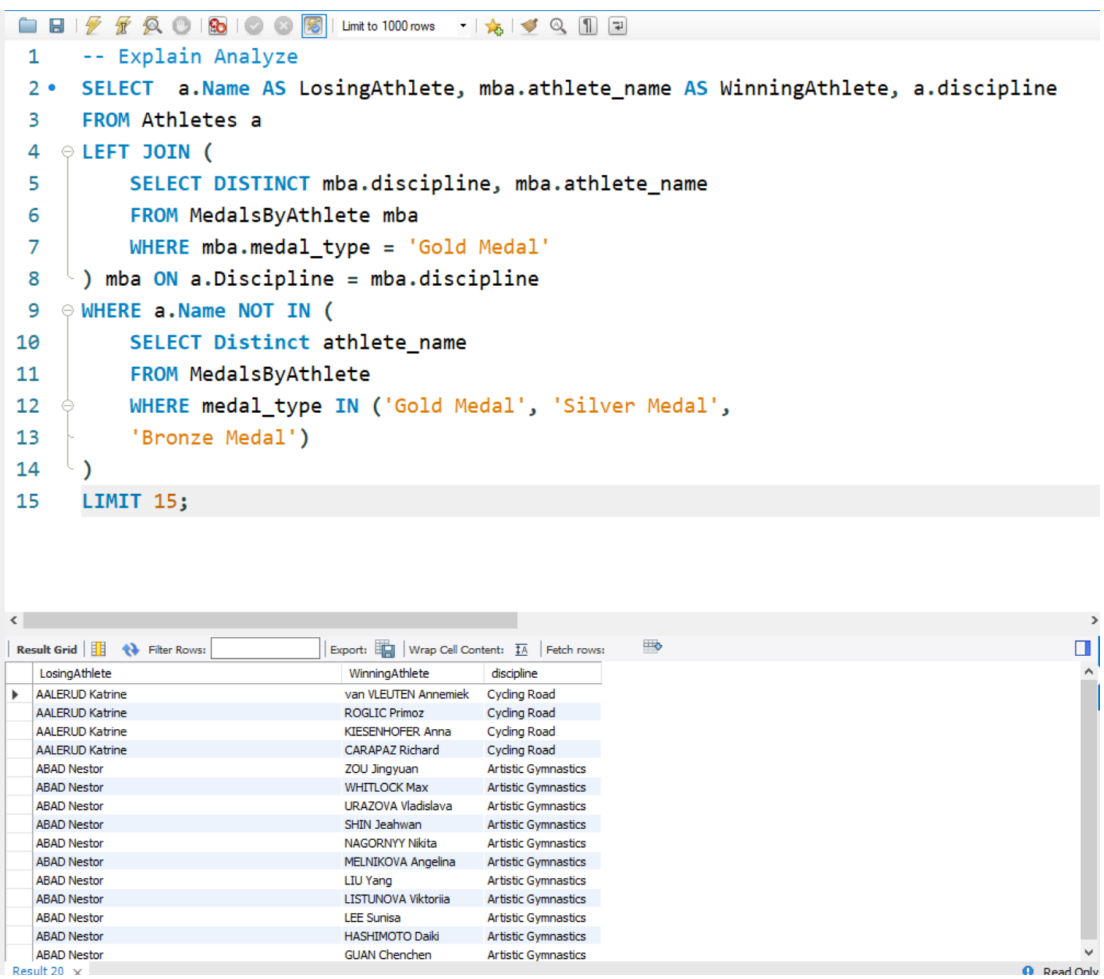
Justification: There are many countries whose coaches do not get recognition for contributing to a nation that wins medals in these competitive games. This advanced query uses INNER JOIN and a subquery. It finds all of the coaches and connects it to the medals table so the

output is only a list of coaches whose athletes have won a gold medal. We also group by the discipline of the athlete and their respective coach's name.

#2 Advanced SQL Query

```
Explain Analyze
SELECT  a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
FROM Athletes a
LEFT JOIN (
    SELECT DISTINCT mba.discipline, mba.athlete_name
    FROM MedalsByAthlete mba
    WHERE mba.medal_type = 'Gold Medal'
) mba ON a.Discipline = mba.discipline
WHERE a.Name NOT IN (
    SELECT Distinct athlete_name
    FROM MedalsByAthlete
    WHERE medal_type IN ('Gold Medal', 'Silver Medal',
    'Bronze Medal')
);
```

Image:



LosingAthlete	WinningAthlete	discipline
AALERUD Katrine	van VLEUTEN Annemiek	Cycling Road
AALERUD Katrine	ROGLIC Primoz	Cycling Road
AALERUD Katrine	KIESENHOFER Anna	Cycling Road
AALERUD Katrine	CARAPAZ Richard	Cycling Road
ABAD Nestor	ZOU Jingyuan	Artistic Gymnastics
ABAD Nestor	WHITLOCK Max	Artistic Gymnastics
ABAD Nestor	URAZOVA Vladislava	Artistic Gymnastics
ABAD Nestor	SHIN Jeahwan	Artistic Gymnastics
ABAD Nestor	NAGORNYY Nikita	Artistic Gymnastics
ABAD Nestor	MELNIKOVA Angelina	Artistic Gymnastics
ABAD Nestor	LIU Yang	Artistic Gymnastics
ABAD Nestor	LISTUNOVA Viktoriia	Artistic Gymnastics
ABAD Nestor	LEE Sunisa	Artistic Gymnastics
ABAD Nestor	HASHIMOTO Daiki	Artistic Gymnastics
ABAD Nestor	GUAN Chenchen	Artistic Gymnastics

Justification: For each athlete that didn’t win a medal, we want to display the athlete that won the gold medal for that discipline.

[Indexing]

Advanced Query 1:

1. Indexes before:

```
show indexes from MedalsByAthlete;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
MedalsByAthlete	0	PRIMARY	1	athlete_name	A	2174	50	NULL		BTREE
MedalsByAthlete	0	PRIMARY	2	event	A	2401	50	NULL		BTREE

Current runtime:

```
Explain analyze
SELECT distinct
    Athletes.Discipline, Coaches.Name
FROM
    Coaches
        INNER JOIN
    Athletes ON Coaches.NOC = Athletes.NOC
        AND Coaches.Discipline = Athletes.Discipline
        INNER JOIN
    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
WHERE
    MedalsByAthlete.medal_type = 'Gold Medal'
GROUP BY Athletes.Discipline, Coaches.Name
LIMIT 15;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Edit Data for EXPLAIN (VARCHAR)

Binary | Text

```
-> Limit: 15 row(s)  (cost=9785.64..9785.85 rows=15) (actual time=9.399..9.402 rows=15 loops=1)
  -> Table scan on <temporary>  (cost=0.02..13.91 rows=914) (actual time=0.002..0.003 rows=15 loops=1)
    -> Temporary table with deduplication  (cost=9785.64..9799.54 rows=914) (actual time=9.399..9.401 rows=42 loops=1)
      -> Filter: ((Coaches.Discipline = Athletes.Discipline) and (Coaches.NOC = Athletes.NOC))  (cost=9694.24 rows=914) (actual time=7.914..8.643 rows=714 loops=1)
        -> Inner hash join (<hash>(Coaches.Discipline)=<hash>(Athletes.Discipline)), (<hash>(Coaches.NOC)=<hash>(Athletes.NOC))  (cost=9694.24 rows=914) (actual time=7.910..8.272 rows=714 loops=1)
          -> Table scan on Coaches  (cost=0.13 rows=380) (actual time=0.037..0.193 rows=380 loops=1)
          -> Hash
            -> Nested loop inner join  (cost=524.51 rows=240) (actual time=0.095..5.893 rows=715 loops=1)
              -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal')  (cost=263.32 rows=240) (actual time=0.068..1.626 rows=781 loops=1)
                -> Table scan on MedalsByAthlete  (cost=263.32 rows=2401) (actual time=0.060..1.180 rows=2401 loops=1)
              -> Filter: (Athletes.`Name` = MedalsByAthlete.athlete_name)  (cost=0.99 rows=1) (actual time=0.005..0.005 rows=1 loops=781)
```

Data Length: 1568 bytes

New Indexes:

Query 1 | SQL File 1* | SQL File 2* | SQL File 3* | SQL File 4* | SQL File 5* | SQL File 6*

Limit to 1000 rows

```
alter table MedalsByAthlete add index (medal_type(50));
show indexes from MedalsByAthlete;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
MedalsByAthlete	0	PRIMARY	1	athlete_name	A	2174	50
MedalsByAthlete	0	PRIMARY	2	event	A	2401	50
MedalsByAthlete	1	medal_type	1	medal_type	A	3	50

New Runtime:

Limit to 1000 rows

```
Explain analyze
SELECT distinct
    Athletes.Discipline, Coaches.Name
FROM
    Coaches
        INNER JOIN
    Athletes ON Coaches.NOC = Athletes.NOC
        AND Coaches.Discipline = Athletes.Discipline
        INNER JOIN
    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
WHERE
    MedalsByAthlete.medal_type = 'Gold Medal'
GROUP BY Athletes.Discipline, Coaches.Name
LIMIT 15;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Edit Data for EXPLAIN (VARCHAR)

Binary | Text

```
-> Limit: 15 row(s)  (cost=357332.56..357332.74 rows=15) (actual time=839.093..839.099 rows=15 loops=1)
    -> Table scan on <temporary>  (cost=0.01..373.48 rows=29678) (actual time=0.004..0.007 rows=15 loops=1)
        -> Temporary table with deduplication  (cost=357332.56..357706.02 rows=29678) (actual time=839.091..839.096 rows=42 loops=1)
            -> Nested loop inner join  (cost=354364.75 rows=29678) (actual time=0.874..837.181 rows=714 loops=1)
                -> Inner hash join (no condition)  (cost=31555.69 rows=296780) (actual time=0.365..43.008 rows=296780 loops=1)
                    -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal')  (cost=5.04 rows=781) (actual time=0.076..4.315 rows=781 loops=1)
                        -> Index range scan on MedalsByAthlete using medal_type  (cost=5.04 rows=781) (actual time=0.073..3.792 rows=781 loops=1)
                    -> Hash
                        -> Table scan on Coaches  (cost=41.00 rows=380) (actual time=0.048..0.190 rows=380 loops=1)
                -> Filter: ((Athletes.Discipline = Coaches.Discipline) and (Athletes.NOC = Coaches.NOC) and (Athletes.`Name` = MedalsByAthlete.athlete_name))  (cost=0.99 rows=0) (actual time=0.003..0.003 rows=0 loops=296780)
                    -> Single-row index lookup on Athletes using PRIMARY (Name=MedalsByAthlete.athlete_name, Discipline=Coaches.Discipline)  (cost=0.99 rows=1) (actual time=0.002..0.002 rows=0 loops=296780)
```

Data Length: 1448 bytes

Result: Twice as slow

Default Indexing And Runtime = 6.275:

```
show indexes from Coaches;
-- show indexes from MedalsByAthlete;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	I
Coaches	0	PRIMARY	1	Name	A	374	50	NULL		B
Coaches	0	PRIMARY	2	Event	A	380	50	NULL		B

```
-- show indexes from Coaches;
show indexes from MedalsByAthlete;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	N
MedalsByAthlete	0	PRIMARY	1	athlete_name	A	2174	50	NULL	
MedalsByAthlete	0	PRIMARY	2	event	A	2401	50	NULL	

Adding index to Coaches.NOC:

```
alter table Coaches add index (NOC(50));
show indexes from Coaches;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	I
Coaches	0	PRIMARY	1	Name	A	374	50	NULL		B
Coaches	0	PRIMARY	2	Event	A	380	50	NULL		B
Coaches	1	NOC	1	NOC	A	61	50	NULL	YES	B

Runtime:

```sql
Explain analyze
SELECT distinct
    Athletes.Discipline, Coaches.Name
FROM
    Coaches
        INNER JOIN
    Athletes ON Coaches.NOC = Athletes.NOC
        AND Coaches.Discipline = Athletes.Discipline
        INNER JOIN
    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
WHERE
    MedalsByAthlete.medal_type = 'Gold Medal'
GROUP BY Athletes.Discipline, Coaches.Name
LIMIT 15;
```

Edit Data for EXPLAIN (VARCHAR)

Binary | Text

```
-> Limit: 15 row(s)  (cost=5188.25..5188.50 rows=15) (actual time=30.017..30.020 rows=15 loops=1)
    -> Table scan on <temporary>  (cost=0.02..8.59 rows=487) (actual time=0.003..0.005 rows=15 loops=1)
        -> Temporary table with deduplication  (cost=5188.25..5196.82 rows=487) (actual time=30.016..30.018 rows=42 loops=1)
            -> Nested loop inner join  (cost=5139.50 rows=487) (actual time=0.206..29.062 rows=714 loops=1)
                -> Nested loop inner join  (cost=997.38 rows=782) (actual time=0.153..7.139 rows=715 loops=1)
                    -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal')  (cost=147.75 rows=781) (actual time=0.128..2.653 rows=781 loops=1)
                        -> Index lookup on MedalsByAthlete using medal_type (medal_type='Gold Medal')  (cost=147.75 rows=781) (actual time=0.124..2.393 rows=781 loops=1)
                    -> Filter: ((Athletes.`Name` = MedalsByAthlete.athlete_name) and (Athletes.NOC is not null))  (cost=0.99 rows=1) (actual time=0.005..0.006 rows=1 loops=781)
                        -> Index lookup on Athletes using PRIMARY (Name=MedalsByAthlete.athlete_name)  (cost=0.99 rows=1) (actual time=0.004..0.005 rows=1 loops=781)
                -> Filter: ((Coaches.Discipline = Athletes.Discipline) and (Coaches.NOC = Athletes.NOC))  (cost=4.67 rows=1) (actual time=0.024..0.030 rows=1 loops=715)
                    -> Index lookup on Coaches using NOC (NOC=Athletes.NOC)  (cost=4.67 rows=6) (actual time=0.013..0.028 rows=14 loops=715)
```

Result: The original cost of the algorithm was 9785 with default indexing, but by indexing Coaches.NOC, it went down to 1812.

Indexing both MedalsByAthlete.medal_type and Coaches.NOC
Runtime:

```
Explain analyze
SELECT distinct
    Athletes.Discipline, Coaches.Name
FROM
    Coaches
        INNER JOIN
    Athletes ON Coaches.NOC = Athletes.NOC
        AND Coaches.Discipline = Athletes.Discipline
        INNER JOIN
    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
WHERE
    MedalsByAthlete.medal_type = 'Gold Medal'
GROUP BY Athletes.Discipline, Coaches.Name
LIMIT 15;
```

Edit Data for EXPLAIN (VARCHAR)

Binary | Text

```
-> Limit: 15 row(s)  (cost=5188.25..5188.50 rows=15) (actual time=30.017..30.020 rows=15 loops=1)
    -> Table scan on <temporary>  (cost=0.02..8.59 rows=487) (actual time=0.003..0.005 rows=15 loops=1)
        -> Temporary table with deduplication  (cost=5188.25..5196.82 rows=487) (actual time=30.016..30.018 rows=42 loops=1)
            -> Nested loop inner join  (cost=5139.50 rows=487) (actual time=0.206..29.062 rows=714 loops=1)
                -> Nested loop inner join  (cost=997.38 rows=782) (actual time=0.153..7.139 rows=715 loops=1)
                    -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal')  (cost=147.75 rows=781) (actual time=0.128..2.653 rows=781 loops=1)
                        -> Index lookup on MedalsByAthlete using medal_type (medal_type='Gold Medal')  (cost=147.75 rows=781) (actual time=0.124..2.393 rows=781 loops=1)
                    -> Filter: ((Athletes.`Name` = MedalsByAthlete.athlete_name) and (Athletes.NOC is not null))  (cost=0.99 rows=1) (actual time=0.005..0.006 rows=1 loops=781)
                        -> Index lookup on Athletes using PRIMARY (Name=MedalsByAthlete.athlete_name)  (cost=0.99 rows=1) (actual time=0.004..0.005 rows=1 loops=781)
                -> Filter: ((Coaches.Discipline = Athletes.Discipline) and (Coaches.NOC = Athletes.NOC))  (cost=4.67 rows=1) (actual time=0.024..0.030 rows=1 loops=715)
                    -> Index lookup on Coaches using NOC (NOC=Athletes.NOC)  (cost=4.67 rows=6) (actual time=0.013..0.028 rows=14 loops=715)
```

Results: Faster than default indexing, but for some reason adding indexing MedalsByAthlete.medal_type makes the algorithm slower, even when paired with indexing on Coaches.NOC.

We will choose to add index to Coaches.NOC but not index MedalsByAthlete.medal_type as this will produce the lowest runtime.

2.

Advanced Query 2

Default Indexing runtime:

```
Explain Analyze
SELECT  a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
FROM Athletes a
LEFT JOIN (
    SELECT DISTINCT mba.discipline, mba.athlete_name
    FROM MedalsByAthlete mba
    WHERE mba.medal_type = 'Gold Medal'
) mba ON a.Discipline = mba.discipline
WHERE a.Name NOT IN (
    SELECT Distinct athlete_name
    FROM MedalsByAthlete
    WHERE medal_type IN ('Gold Medal', 'Silver Medal',
    'Bronze Medal')
);
```

Edit Data for EXPLAIN (VARCHAR)

Binary | Text

```
-> Left hash join (<hash>(mba.discipline)=<hash>(a.Discipline)), extra conditions: (mba.discipline = a.Discipline)  (cost=644737605.88 rows=6447353534) (actual time=6.277..111.975 rows=297006 loops=1)
    -> Nested loop antijoin  (cost=2687611.01 rows=26852784) (actual time=3.274..16.645 rows=9069 loops=1)
        -> Table scan on a  (cost=1214.21 rows=11184) (actual time=0.033..4.074 rows=11077 loops=1)
        -> Single-row index lookup on <subquery3> using <auto_distinct_key> (athlete_name=a.`Name`)  (actual time=0.001..0.001 rows=0 loops=11077)
            -> Materialize with deduplication  (cost=503.42..503.42 rows=2401) (actual time=10.949..10.949 rows=2174 loops=1)
                -> Filter: (MedalsByAthlete.athlete_name is not null)  (cost=263.32 rows=2401) (actual time=0.030..2.220 rows=2401 loops=1)
                    -> Filter: (MedalsByAthlete.medal_type in ('Gold Medal','Silver Medal','Bronze Medal'))  (cost=263.32 rows=2401) (actual time=0.028..2.036 rows=2401 loops=1)
                        -> Table scan on MedalsByAthlete  (cost=263.32 rows=2401) (actual time=0.025..0.828 rows=2401 loops=1)
    -> Hash
        -> Table scan on mba  (cost=0.02..5.50 rows=240) (actual time=0.001..0.059 rows=735 loops=1)
            -> Materialize  (cost=316.86..322.34 rows=240) (actual time=2.672..2.772 rows=735 loops=1)
                -> Table scan on <temporary>  (cost=0.02..5.50 rows=240) (actual time=0.002..0.081 rows=735
```

Data Length: 1847 bytes

Save... | Close

Adding Index on MedalsByAthlete.discipline:

```
-- alter table Coaches add index (NOC(50));
alter table MedalsByAthlete add index (discipline(50));
show indexes from MedalsByAthlete;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
MedalsByAthlete	0	PRIMARY	1	athlete_name	A	2174	50	NULL		BTREE
MedalsByAthlete	0	PRIMARY	2	event	A	2401	50	NULL		BTREE
MedalsByAthlete	1	discipline	1	discipline	A	46	50	NULL	YES	BTREE

Runtime:

```
Explain Analyze
SELECT  a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
FROM Athletes a
LEFT JOIN (
    SELECT DISTINCT mba.discipline, mba.athlete_name
    FROM MedalsByAthlete mba
    WHERE mba.medal_type = 'Gold Medal'
) mba ON a.Discipline = mba.discipline
WHERE a.Name NOT IN (
    SELECT Distinct athlete_name
    FROM MedalsByAthlete
    WHERE medal_type IN ('Gold Medal', 'Silver Medal',
    'Bronze Medal')
)
```

Edit Data for EXPLAIN (VARCHAR)

```
-> Limit: 15 row(s)  (cost=644737605.88 rows=15) (actual time=8.137..8.151 rows=15 loops=1)
  -> Left hash join (<hash>(mba.discipline)=<hash>(a.Discipline)), extra conditions: (mba.discipline = a.Discipline)  (cost=644737605.88 rows=6447353534) (actual time=8.136..8.149 rows=15 loops=1)
      -> Nested loop antijoin  (cost=2687611.01 rows=26852784) (actual time=4.290..4.294 rows=2 loops=1)
          -> Table scan on a  (cost=1214.21 rows=11184) (actual time=0.045..0.046 rows=2 loops=1)
          -> Single-row index lookup on <subquery3> using <auto_distinct_key> (athlete_name=a.`Name`) (actual time=0.003..0.003 rows=0 loops=2)
              -> Materialize with deduplication  (cost=503.42..503.42 rows=2401) (actual time=4.245..4.245 rows=2174 loops=1)
                  -> Filter: (MedalsByAthlete.athlete_name is not null)  (cost=263.32 rows=2401) (actual time=0.052..2.924 rows=2401 loops=1)
                      -> Filter: (MedalsByAthlete.medal_type in ('Gold Medal','Silver Medal','Bronze Medal'))  (cost=263.32 rows=2401) (actual time=0.050..2.673 rows=2401 loops=1)
                          -> Table scan on MedalsByAthlete  (cost=263.32 rows=2401) (actual time=0.046..1.096 rows=2401 loops=1)
      -> Hash
          -> Table scan on mba  (cost=0.02..5.50 rows=240) (actual time=0.002..0.082 rows=735 loops=1)
              -> Materialize  (cost=316.86..322.34 rows=240) (actual time=3.327..3.463 rows=735 loops=1)
```

Data Length: 1979 bytes

Results: Less Efficient, the actual time went from 6.277 - 8.137

Indexing medal_type instead of discipline:

```
alter table MedalsByAthlete add index (medal_type(50));
show indexes from MedalsByAthlete;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Co
MedalsByAthlete	0	PRIMARY	1	athlete_name	A	2174	50	NULL		BTREE	
MedalsByAthlete	0	PRIMARY	2	event	A	2401	50	NULL		BTREE	
MedalsByAthlete	1	medal_type	1	medal_type	A	3	50	NULL	YES	BTREE	

Runtime:

```
Explain Analyze
SELECT  a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
FROM Athletes a
LEFT JOIN (
    SELECT DISTINCT mba.discipline, mba.athlete_name
    FROM MedalsByAthlete mba
    WHERE mba.medal_type = 'Gold Medal'
) mba ON a.Discipline = mba.discipline
WHERE a.Name NOT IN (
    SELECT Distinct athlete_name
    FROM MedalsByAthlete
    WHERE medal_type IN ('Gold Medal', 'Silver Medal',
    'Bronze Medal')
)
LIMIT 15;
```

Edit Data for EXPLAIN (VARCHAR)

Binary | Text

```
-> Limit: 15 row(s)  (cost=644737605.88 rows=15) (actual time=8.137..8.151 rows=15 loops=1)
    -> Left hash join (<hash>(mba.discipline)=<hash>(a.Discipline)), extra conditions: (mba.discipline = a.Discipline)  (cost=644737605.88 rows=6447353534) (actual time=8.136..8.149 rows=15 loops=1)
        -> Nested loop antijoin  (cost=2687611.01 rows=26852784) (actual time=4.290..4.294 rows=2 loops=1)
            -> Table scan on a  (cost=1214.21 rows=11184) (actual time=0.045..0.046 rows=2 loops=1)
            -> Single-row index lookup on <subquery3> using <auto_distinct_key> (athlete_name=a.`Name`)  (actual time=0.003..0.003 rows=0 loops=2)
                -> Materialize with deduplication  (cost=503.42..503.42 rows=2401) (actual time=4.245..4.245 rows=2174 loops=1)
                    -> Filter: (MedalsByAthlete.athlete_name is not null)  (cost=263.32 rows=2401) (actual time=0.052..2.924 rows=2401 loops=1)
                        -> Filter: (MedalsByAthlete.medal_type in ('Gold Medal','Silver Medal','Bronze Medal'))  (cost=263.32 rows=2401) (actual time=0.050..2.673 rows=2401 loops=1)
                            -> Table scan on MedalsByAthlete  (cost=263.32 rows=2401) (actual time=0.046..1.096 rows=2401 loops=1)
        -> Hash
            -> Table scan on mba  (cost=0.02..5.50 rows=240) (actual time=0.002..0.082 rows=735 loops=1)
                -> Materialize  (cost=316.86..322.34 rows=240) (actual time=3.327..3.463 rows=735 loops=1)
```

Data Length: 1979 bytes

Save... | Close

Read Only

Results:

Still slower than the default indexing, which is weird since the query has two subquerys that use the ‘WHERE clause’

Indexing both discipline and medal_type:

```
alter table MedalsByAthlete add index (discipline(50));
show indexes from MedalsByAthlete;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
MedalsByAthlete	0	PRIMARY	1	athlete_name	A	2174	50	NULL		BTREE			YES
MedalsByAthlete	0	PRIMARY	2	event	A	2401	50	NULL		BTREE			YES
MedalsByAthlete	1	medal_type	1	medal_type	A	3	50	NULL	YES	BTREE			YES
MedalsByAthlete	1	discipline	1	discipline	A	46	50	NULL	YES	BTREE			YES

Runtime:

```
Explain Analyze
SELECT  a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
FROM Athletes a
LEFT JOIN (
    SELECT DISTINCT mba.discipline, mba.athlete_name
    FROM MedalsByAthlete mba
    WHERE mba.medal_type = 'Gold Medal'
) mba ON a.Discipline = mba.discipline
WHERE a.Name NOT IN (
    SELECT Distinct athlete_name
    FROM MedalsByAthlete
    WHERE medal_type IN ('Gold Medal', 'Silver Medal',
    'Bronze Medal')
)
```

Edit Data for EXPLAIN (VARCHAR)

Binary | Text

```
-> Left hash join (<hash>(mba.discipline)=<hash>(a.Discipline)), extra conditions: (mba.discipline = a.Discipline)  (cost=2097207062.12 rows=20972024304) (actual time=6.878..104.717 rows=297006 loops=1)
    -> Nested loop antijoin  (cost=2687611.01 rows=26852784) (actual time=3.109..15.352 rows=9069 loops=1)
        -> Table scan on a  (cost=1214.21 rows=11184) (actual time=0.043..3.973 rows=11077 loops=1)
        -> Single-row index lookup on <subquery3> using <auto_distinct_key> (athlete_name=a.`Name`)  (actual time=0.000..0.000 rows=0 loops=11077)
            -> Materialize with deduplication  (cost=503.42..503.42 rows=2401) (actual time=9.884..9.884 rows=2174 loops=1)
                -> Filter: (MedalsByAthlete.athlete_name is not null)  (cost=263.32 rows=2401) (actual time=0.032..2.126 rows=2401 loops=1)
                    -> Filter: (MedalsByAthlete.medal_type in ('Gold Medal','Silver Medal','Bronze Medal'))  (cost=263.32 rows=2401) (actual time=0.031..1.931 rows=2401 loops=1)
                        -> Table scan on MedalsByAthlete  (cost=263.32 rows=2401) (actual time=0.028..0.791 rows=2401 loops=1)
    -> Hash
        -> Table scan on mba  (cost=0.02..12.26 rows=781) (actual time=0.002..0.061 rows=735 loops=1)
            -> Materialize  (cost=316.23..328.48 rows=781) (actual time=3.411..3.520 rows=735 loops=1)
                -> Table scan on <temporary>  (cost=0.02..12.26 rows=781) (actual time=0.002..0.077 rows=735
```

Data Length: 1892 bytes

Save... Close

Read Only

Results:

By using both discipline and medal_type as an index, we were not able to speedup the runtime of the query with both versions costing around 6.8 units of time.

Based on these results, we will stick with default indexing as this produces the shortest runtime.