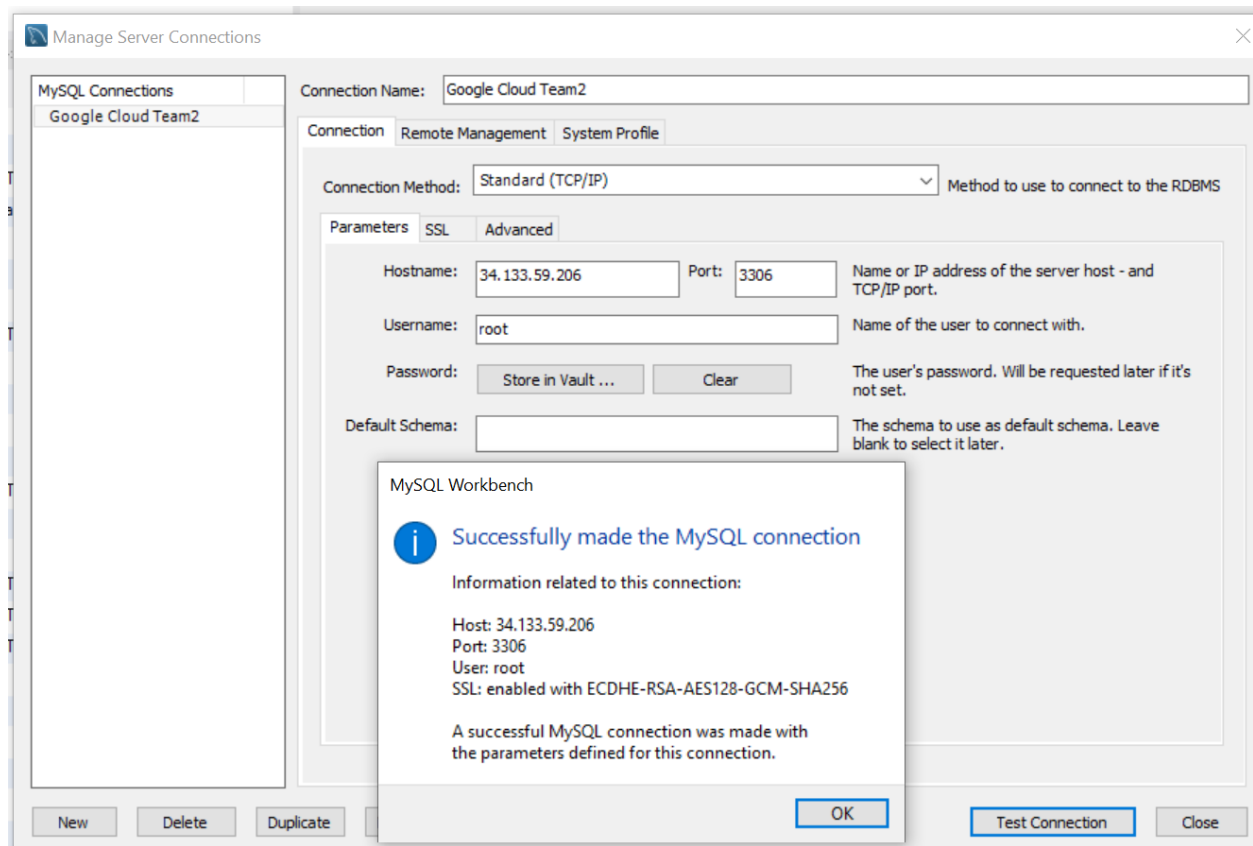


A. Create a markdown or pdf called “Database Design” in the doc folder

GCP Connection:



DDL for Table Creation:

```
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:

The screenshot shows a database query tool interface. The SQL query entered is `select count(*) from Athletes;`. The result is displayed in a table with one row and one column, showing the count of 11077.

count(*)
11077

Coaches:

The screenshot shows a database query tool interface. The SQL query entered is `select count(*) from Coaches;`. The result is displayed in a table with one row and one column, showing the count of 380.

count(*)
380

Medals:

Limit to 1000 rows

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

2

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

count(*)
93

MedalsByAthlete:

Limit to 1000 rows

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

2

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

count(*)
2401

Officials:

Limit to 1000 rows

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

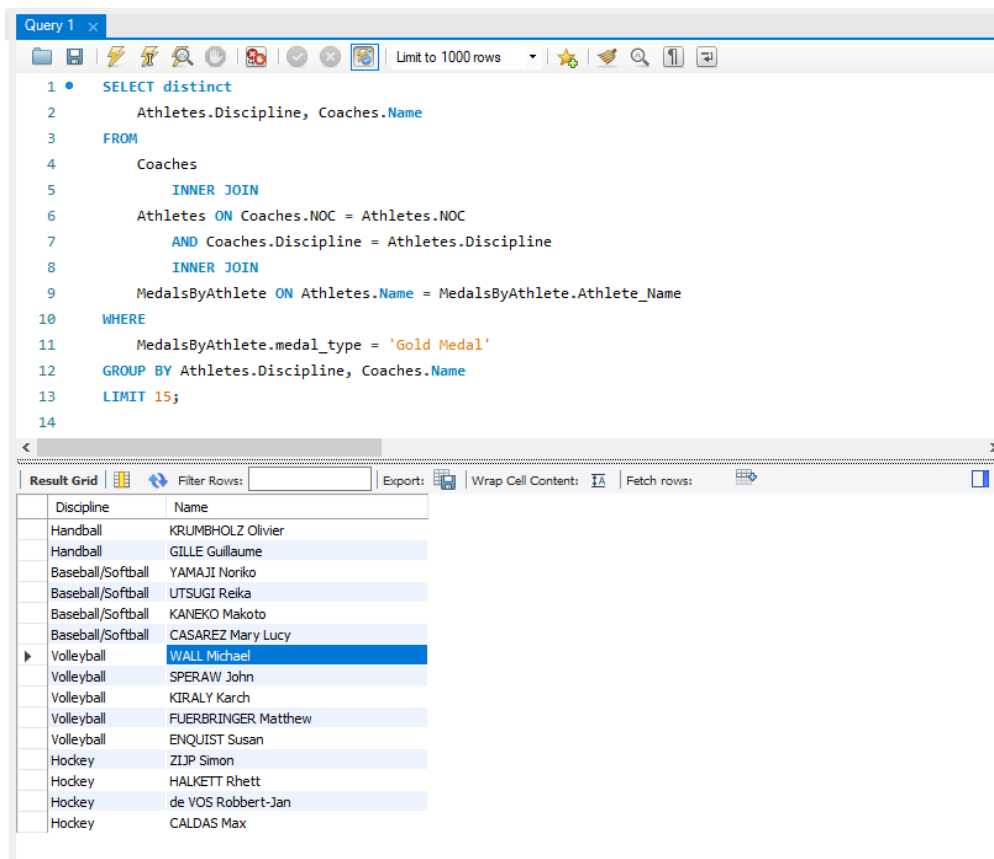
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

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:



The screenshot shows a SQL query editor with the following query:

```
1 SELECT distinct
2   Athletes.Discipline, Coaches.Name
3 FROM
4   Coaches
5   INNER JOIN
6     Athletes ON Coaches.NOC = Athletes.NOC
7     AND Coaches.Discipline = Athletes.Discipline
8   INNER JOIN
9     MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
10 WHERE
11   MedalsByAthlete.medal_type = 'Gold Medal'
12 GROUP BY Athletes.Discipline, Coaches.Name
13 LIMIT 15;
```

The results are displayed in a table with the following data:

Discipline	Name
Handball	KRUMBHOLZ Olivier
Handball	GILLE Guillaume
Baseball/Softball	YAMAJI Noriko
Baseball/Softball	UTSUGI Reika
Baseball/Softball	KANEKO Makoto
Baseball/Softball	CASAREZ Mary Lucy
Volleyball	WALL Michael
Volleyball	SPERAW John
Volleyball	KIRALY Karch
Volleyball	FUERBRINGER Matthew
Volleyball	ENQUIST Susan
Hockey	ZIJP Simon
Hockey	HALKETT Rhett
Hockey	de VOS Robbert-Jan
Hockey	CALDAS Max

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:

The screenshot shows a SQL query editor with a query window and a results window. The query window contains the following SQL code:

```
1  -- Explain Analyze
2  • SELECT a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
3  FROM Athletes a
4  LEFT JOIN (
5      SELECT DISTINCT mba.discipline, mba.athlete_name
6      FROM MedalsByAthlete mba
7      WHERE mba.medal_type = 'Gold Medal'
8  ) mba ON a.Discipline = mba.discipline
9  WHERE a.Name NOT IN (
10     SELECT Distinct athlete_name
11     FROM MedalsByAthlete
12     WHERE medal_type IN ('Gold Medal', 'Silver Medal',
13     'Bronze Medal')
14 )
15 LIMIT 15;
```

The results window shows a table with three columns: LosingAthlete, WinningAthlete, and discipline. The table contains 15 rows of data, with the first row highlighted. The results are as follows:

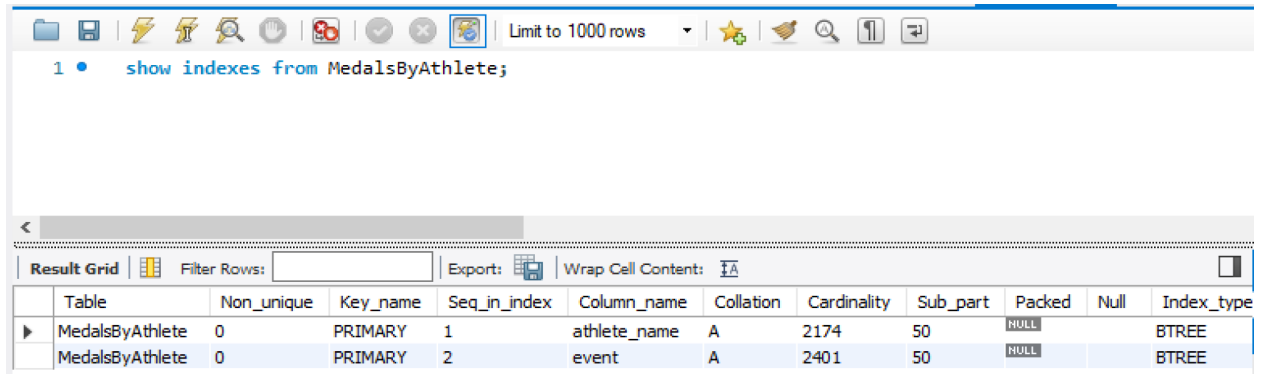
LosingAthlete	WinningAthlete	discipline
AALERUD Katrine	van VLEUTEN Annemiek	Cycling Road
AALERUD Katrine	ROGLIC Primoz	Cycling Road
AALERUD Katrine	KIESENHOFER Anna	Cycling Road
AALERUD Katrine	CARAPAZ Richard	Cycling Road
ABAD Nestor	ZOU Jingyuan	Artistic Gymnastics
ABAD Nestor	WHITLOCK Max	Artistic Gymnastics
ABAD Nestor	URAZOVA Vladislava	Artistic Gymnastics
ABAD Nestor	SHIN Jaehwan	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:



1 • `show indexes from MedalsByAthlete;`

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [iA](#)

	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:

The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```
1 • Explain analyze
2 SELECT distinct
3     Athletes.Discipline, Coaches.Name
4 FROM
5     Coaches
6     INNER JOIN
7     Athletes ON Coaches.NOC = Athletes.NOC
8     AND Coaches.Discipline = Athletes.Discipline
9     INNER JOIN
10    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
11 WHERE
12     MedalsByAthlete.medal_type = 'Gold Medal'
13 GROUP BY Athletes.Discipline, Coaches.Name
14 LIMIT 15;
15
```

Below the query editor, there is a 'Result Grid' section with options for 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below that is a window titled 'Edit Data for EXPLAIN (VARCHAR)' with tabs for 'Binary' and 'Text'. The 'Text' tab is selected, showing the execution plan:

```
1 -> Limit: 15 row(s) (cost=9785.64..9785.85 rows=15) (actual time=9.399..9.402 rows=15 loops=1)
2 -> Table scan on <temporary> (cost=0.02..13.91 rows=914) (actual time=0.002..0.003 rows=15 loops=1)
3 -> Temporary table with deduplication (cost=9785.64..9799.54 rows=914) (actual time=9.399..9.401
rows=42 loops=1)
4 -> 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)
5 -> 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)
6 -> Table scan on Coaches (cost=0.13 rows=380) (actual time=0.037..0.193 rows=380 loops=1)
7 -> Hash
8 -> Nested loop inner join (cost=524.51 rows=240) (actual time=0.095..5.893 rows=715 loops=1)
9 -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal') (cost=263.32 rows=240) (actual
time=0.068..1.626 rows=781 loops=1)
10 -> Table scan on MedalsByAthlete (cost=263.32 rows=2401) (actual time=0.060..1.180
rows=2401 loops=1)
11 -> 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* x SQL File 3* SQL File 4* SQL File 5* SQL File 6*

1 • `alter table MedalsByAthlete add index (medal_type(50));`
2 • `show indexes from MedalsByAthlete;`
3

Result Grid

	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:

1 • `Explain analyze`
2 `SELECT distinct`
3 `Athletes.Discipline, Coaches.Name`
4 `FROM`
5 `Coaches`
6 `INNER JOIN`
7 `Athletes ON Coaches.NOC = Athletes.NOC`
8 `AND Coaches.Discipline = Athletes.Discipline`
9 `INNER JOIN`
10 `MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name`
11 `WHERE`
12 `MedalsByAthlete.medal_type = 'Gold Medal'`
13 `GROUP BY Athletes.Discipline, Coaches.Name`
14 `LIMIT 15;`
15

Result Grid

Edit Data for EXPLAIN (VARCHAR)

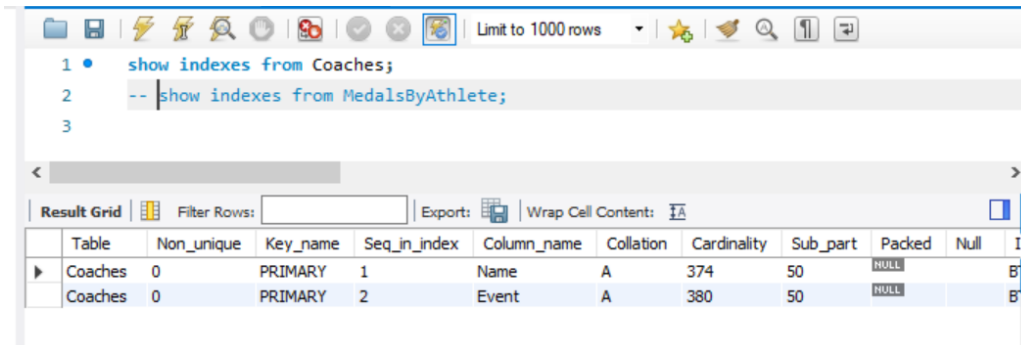
Binary Text

```
1 -> Limit: 15 row(s) (cost=357332.56..357332.74 rows=15) (actual time=839.093..839.099 rows=15 loops=1)
2 -> Table scan on <temporary> (cost=0.01..373.48 rows=29678) (actual time=0.004..0.007 rows=15 loops=1)
3 -> Temporary table with deduplication (cost=357332.56..357706.02 rows=29678) (actual
time=839.091..839.096 rows=42 loops=1)
4 -> Nested loop inner join (cost=354364.75 rows=29678) (actual time=0.874..837.181 rows=714 loops=1)
5 -> Inner hash join (no condition) (cost=31555.69 rows=29678) (actual time=0.365..43.008
rows=29678) loops=1)
6 -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal') (cost=5.04 rows=781) (actual
time=0.076..4.315 rows=781 loops=1)
7 -> Index range scan on MedalsByAthlete using medal_type (cost=5.04 rows=781) (actual
time=0.073..3.792 rows=781 loops=1)
8 -> Hash
9 -> Table scan on Coaches (cost=41.00 rows=380) (actual time=0.048..0.190 rows=380 loops=1)
10 -> 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=29678)
11 -> 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=29678)
```

Data Length: 1448 bytes

Result: Twice as slow

Default Indexing And Runtime = 6.275:

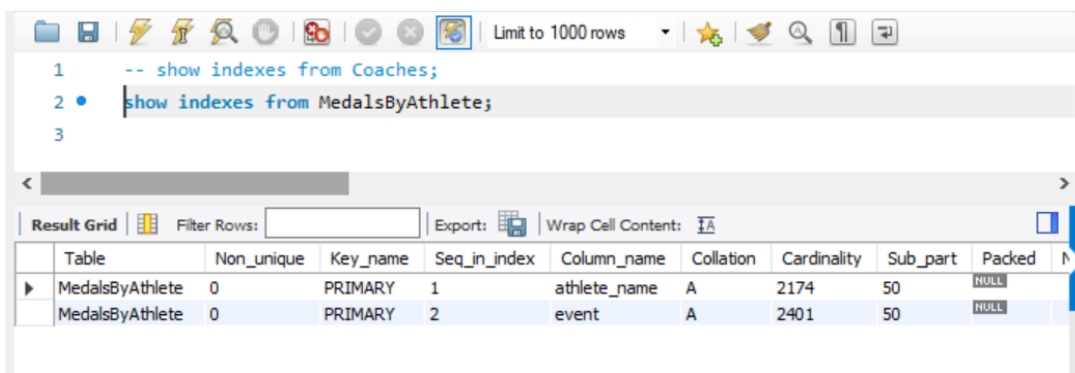


SQL Command:

```
1 • show indexes from Coaches;  
2 -- show indexes from MedalsByAthlete;  
3
```

Result Grid:

	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



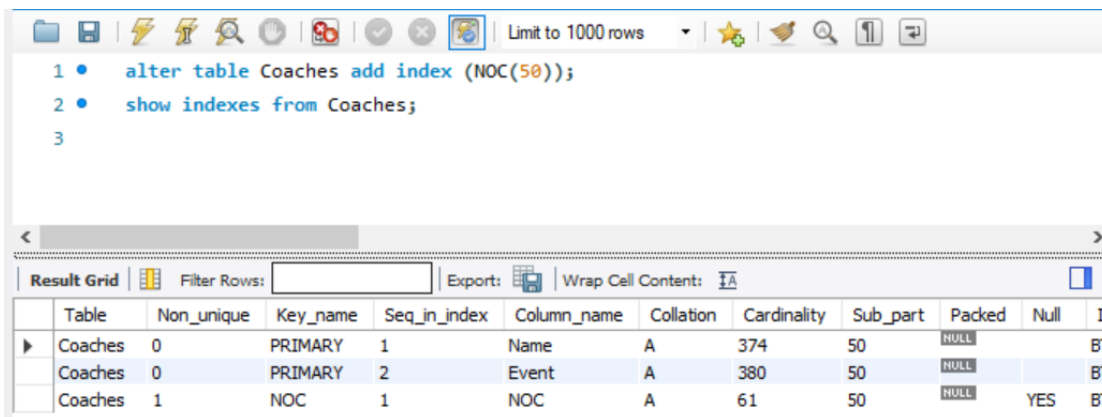
SQL Command:

```
1 -- show indexes from Coaches;  
2 • show indexes from MedalsByAthlete;  
3
```

Result Grid:

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

Adding index to Coaches.NOC:



SQL Command:

```
1 • alter table Coaches add index (NOC(50));  
2 • show indexes from Coaches;  
3
```

Result Grid:

	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:

The screenshot shows a database IDE with a SQL query editor and an execution plan window. The query is as follows:

```
1 • Explain analyze
2 SELECT distinct
3     Athletes.Discipline, Coaches.Name
4 FROM
5     Coaches
6     INNER JOIN
7     Athletes ON Coaches.NOC = Athletes.NOC
8     AND Coaches.Discipline = Athletes.Discipline
9     INNER JOIN
10    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
11 WHERE
12     MedalsByAthlete.medal_type = 'Gold Medal'
13 GROUP BY Athletes.Discipline, Coaches.Name
14 LIMIT 15;
```

The execution plan window, titled "Edit Data for EXPLAIN (VARCHAR)", shows the following details:

```
1 -> Limit: 15 row(s) (cost=5188.25..5188.50 rows=15) (actual time=30.017..30.020 rows=15 loops=1)
2 -> Table scan on <temporary> (cost=0.02..8.59 rows=487) (actual time=0.003..0.005 rows=15 loops=1)
3 -> Temporary table with deduplication (cost=5188.25..5196.82 rows=487) (actual time=30.016..30.018
   rows=42 loops=1)
4 -> Nested loop inner join (cost=5139.50 rows=487) (actual time=0.206..29.062 rows=714 loops=1)
5 -> Nested loop inner join (cost=997.38 rows=782) (actual time=0.153..7.139 rows=715 loops=1)
6 -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal') (cost=147.75 rows=781) (actual
   time=0.128..2.653 rows=781 loops=1)
7 -> 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)
8 -> 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)
9 -> 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)
10 -> 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)
11 -> 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:

The screenshot shows a database query editor with a SQL query and its execution plan.

SQL Query:

```

1 • Explain analyze
2 SELECT distinct
3     Athletes.Discipline, Coaches.Name
4 FROM
5     Coaches
6     INNER JOIN
7     Athletes ON Coaches.NOC = Athletes.NOC
8     AND Coaches.Discipline = Athletes.Discipline
9     INNER JOIN
10    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
11 WHERE
12    MedalsByAthlete.medal_type = 'Gold Medal'
13 GROUP BY Athletes.Discipline, Coaches.Name
14 LIMIT 15;
15

```

Execution Plan:

```

1  -> Limit: 15 row(s) (cost=5188.25..5188.50 rows=15) (actual time=30.017..30.020 rows=15 loops=1)
2  -> Table scan on <temporary> (cost=0.02..8.59 rows=487) (actual time=0.003..0.005 rows=15 loops=1)
3  -> Temporary table with deduplication (cost=5188.25..5196.82 rows=487) (actual time=30.016..30.018
    rows=42 loops=1)
4  -> Nested loop inner join (cost=5139.50 rows=487) (actual time=0.206..29.062 rows=714 loops=1)
5  -> Nested loop inner join (cost=997.38 rows=782) (actual time=0.153..7.139 rows=715 loops=1)
6  -> Filter: (MedalsByAthlete.medal_type = 'Gold Medal') (cost=147.75 rows=781) (actual
    time=0.128..2.653 rows=781 loops=1)
7  -> 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)
8  -> 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)
9  -> 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)
10 -> 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)
11 -> 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:

The screenshot displays a SQL query in a text editor and its corresponding execution plan in a separate window.

SQL Query:

```
1 • Explain Analyze
2 SELECT a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
3 FROM Athletes a
4 LEFT JOIN (
5     SELECT DISTINCT mba.discipline, mba.athlete_name
6     FROM MedalsByAthlete mba
7     WHERE mba.medal_type = 'Gold Medal'
8 ) mba ON a.Discipline = mba.discipline
9 WHERE a.Name NOT IN (
10     SELECT Distinct athlete_name
11     FROM MedalsByAthlete
12     WHERE medal_type IN ('Gold Medal', 'Silver Medal',
13                          'Bronze Medal')
14 );
```

Execution Plan (Text View):

```
1 -> 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)
2 -> Nested loop antijoin (cost=2687611.01 rows=26852784) (actual time=3.274..16.645 rows=9069 loops=1)
3 -> Table scan on a (cost=1214.21 rows=11184) (actual time=0.033..4.074 rows=11077 loops=1)
4 -> Single-row index lookup on <subquery3> using <auto_distinct_key> (athlete_name=a.'Name') (actual
time=0.001..0.001 rows=0 loops=11077)
5 -> Materialize with deduplication (cost=503.42..503.42 rows=2401) (actual time=10.949..10.949
rows=2174 loops=1)
6 -> Filter: (MedalsByAthlete.athlete_name is not null) (cost=263.32 rows=2401) (actual
time=0.030..2.220 rows=2401 loops=1)
7 -> 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)
8 -> Table scan on MedalsByAthlete (cost=263.32 rows=2401) (actual time=0.025..0.828
rows=2401 loops=1)
9 -> Hash
10 -> Table scan on mba (cost=0.02..5.50 rows=240) (actual time=0.001..0.059 rows=735 loops=1)
11 -> Materialize (cost=316.86..322.34 rows=240) (actual time=2.672..2.772 rows=735 loops=1)
12 -> Table scan on <temporary> (cost=0.02..5.50 rows=240) (actual time=0.002..0.081 rows=735
```

Data Length: 1847 bytes

Buttons: Save... Close

Adding Index on MedalsByAthlete.discipline:

1	-- alter table Coaches add index (NOC(50));
2	• alter table MedalsByAthlete add index (discipline(50));
3	• show indexes from MedalsByAthlete;
4	

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:

```

1 • Explain Analyze
2 SELECT a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
3 FROM Athletes a
4 LEFT JOIN (
5     SELECT DISTINCT mba.discipline, mba.athlete_name
6     FROM MedalsByAthlete mba
7     WHERE mba.medal_type = 'Gold Medal'
8 ) mba ON a.Discipline = mba.discipline
9 WHERE a.Name NOT IN (
10     SELECT Distinct athlete_name
11     FROM MedalsByAthlete
12     WHERE medal_type IN ('Gold Medal', 'Silver Medal',
13                          'Bronze Medal')
14 )

```


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

Edit Data for EXPLAIN (VARCHAR)

Binary Text

```

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

```

Data Length: 1979 bytes

Save... Close

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

Indexing medal_type instead of discipline:

Limit to 1000 rows

```

1 • alter table MedalsByAthlete add index (medal_type(50));
2 • show indexes from MedalsByAthlete;
3

```

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	medal_type	1	medal_type	A	3	50	NULL	YES	BTREE

Runtime:

```

1 • Explain Analyze
2 SELECT a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
3 FROM Athletes a
4 LEFT JOIN (
5     SELECT DISTINCT mba.discipline, mba.athlete_name
6     FROM MedalsByAthlete mba
7     WHERE mba.medal_type = 'Gold Medal'
8 ) mba ON a.Discipline = mba.discipline
9 WHERE a.Name NOT IN (
10     SELECT Distinct athlete_name
11     FROM MedalsByAthlete
12     WHERE medal_type IN ('Gold Medal', 'Silver Medal',
13     'Bronze Medal')
14 )
15 LIMIT 15;

```

Edit Data for EXPLAIN (VARCHAR)

	Binary	Text
1		-> Limit: 15 row(s) (cost=644737605.88 rows=15) (actual time=8.137..8.151 rows=15 loops=1)
2		-> 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)
3		-> Nested loop antijoin (cost=2687611.01 rows=26852784) (actual time=4.290..4.294 rows=2 loops=1)
4		-> Table scan on a (cost=1214.21 rows=11184) (actual time=0.045..0.046 rows=2 loops=1)
5		-> Single-row index lookup on <subquery3> using <auto_distinct_key> (athlete_name=a.`Name`) (actual time=0.003..0.003 rows=0 loops=2)
6		-> Materialize with deduplication (cost=503.42..503.42 rows=2401) (actual time=4.245..4.245 rows=2174 loops=1)
7		-> Filter: (MedalsByAthlete.athlete_name is not null) (cost=263.32 rows=2401) (actual time=0.052..2.924 rows=2401 loops=1)
8		-> 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)
9		-> Table scan on MedalsByAthlete (cost=263.32 rows=2401) (actual time=0.046..1.096 rows=2401 loops=1)
10		-> Hash
11		-> Table scan on mba (cost=0.02..5.50 rows=240) (actual time=0.002..0.082 rows=735 loops=1)
12		-> Materialize (cost=316.86..322.34 rows=240) (actual time=3.327..3.463 rows=735 loops=1)

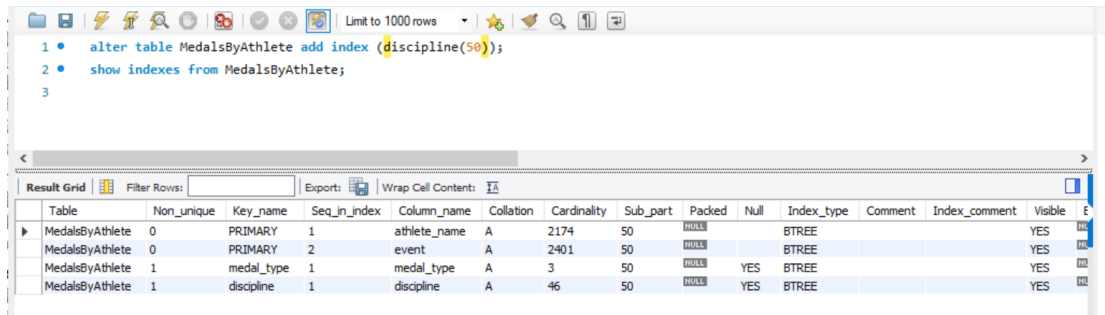
Data Length: 1979 bytes

Save... Close

Results:

Still slower than the default indexing, which is weird since the query has two subqueries that use the 'WHERE clause'

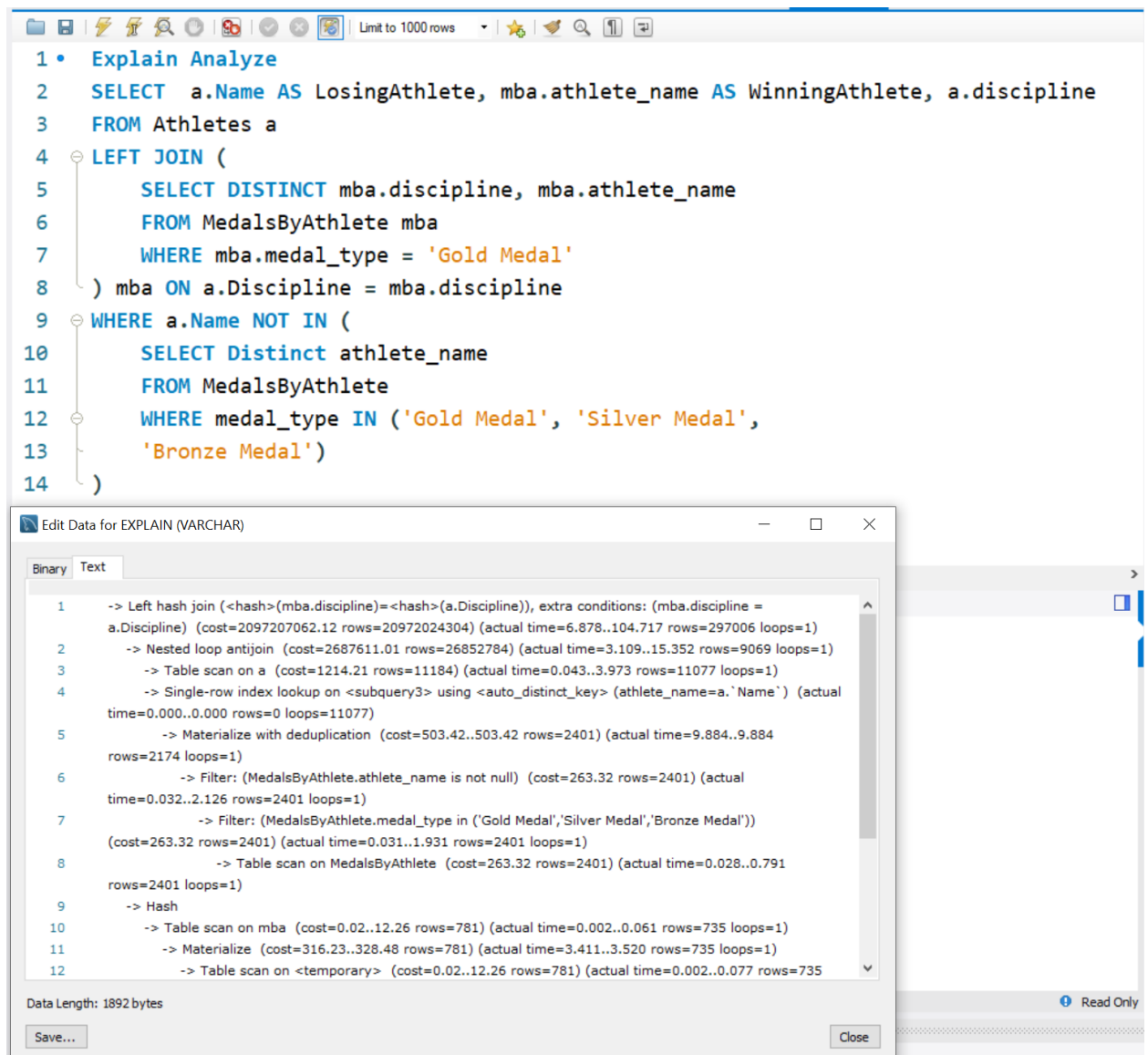
Indexing both discipline and medal_type:



```
1 • alter table MedalsByAthlete add index (discipline(50));
2 • show indexes from MedalsByAthlete;
3
```

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	HULL		BTREE			YES
MedalsByAthlete	0	PRIMARY	2	event	A	2401	50	HULL		BTREE			YES
MedalsByAthlete	1		1	medal_type	A	3	50	HULL	YES	BTREE			YES
MedalsByAthlete	1		1	discipline	A	46	50	HULL	YES	BTREE			YES

Runtime:



```
1 • Explain Analyze
2 SELECT a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
3 FROM Athletes a
4 LEFT JOIN (
5     SELECT DISTINCT mba.discipline, mba.athlete_name
6     FROM MedalsByAthlete mba
7     WHERE mba.medal_type = 'Gold Medal'
8 ) mba ON a.Discipline = mba.discipline
9 WHERE a.Name NOT IN (
10     SELECT Distinct athlete_name
11     FROM MedalsByAthlete
12     WHERE medal_type IN ('Gold Medal', 'Silver Medal',
13                          'Bronze Medal')
14 )
```

Step	Operation	Cost	Rows	Actual Time	Actual Rows	Loops
1	-> Left hash join (<hash>(mba.discipline)=<hash>(a.Discipline)), extra conditions: (mba.discipline = a.Discipline)	2097207062.12	20972024304	6.878..104.717	297006	1
2	-> Nested loop antijoin	2687611.01	26852784	3.109..15.352	9069	1
3	-> Table scan on a	1214.21	11184	0.043..3.973	11077	1
4	-> Single-row index lookup on <subquery3> using <auto_distinct_key> (athlete_name=a.'Name')	0.000..0.000	0		11077	
5	-> Materialize with deduplication	503.42..503.42	2401	9.884..9.884		1
6	-> Filter: (MedalsByAthlete.athlete_name is not null)	263.32	2401	0.032..2.126	2401	1
7	-> Filter: (MedalsByAthlete.medal_type in ('Gold Medal','Silver Medal','Bronze Medal'))	263.32	2401	0.031..1.931	2401	1
8	-> Table scan on MedalsByAthlete	263.32	2401	0.028..0.791		1
9	-> Hash					
10	-> Table scan on mba	0.02..12.26	781	0.002..0.061	735	1
11	-> Materialize	316.23..328.48	781	3.411..3.520	735	1
12	-> Table scan on <temporary>	0.02..12.26	781	0.002..0.077	735	

Data Length: 1892 bytes

Save... Close

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.