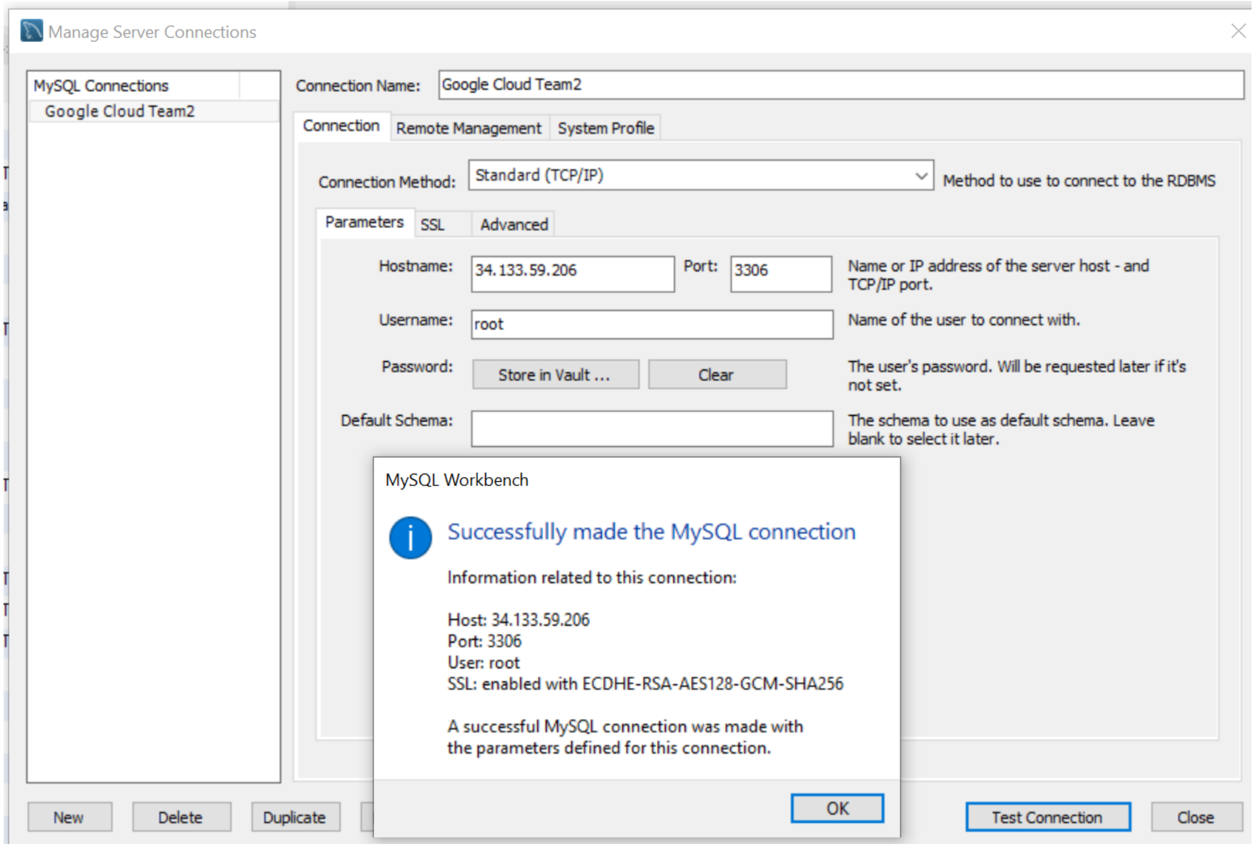
A. Create a markdown or pdf called “Database Design” in the doc folder

GCP Connection:



Connection Name: Google Cloud Team2

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Hostname: 34.133.59.206 Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: root Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

MySQL Workbench

Successfully made the MySQL connection

Information related to this connection:

Host: 34.133.59.206
Port: 3306
User: root
SSL: enabled with ECDHE-RSA-AES128-GCM-SHA256

A successful MySQL connection was made with the parameters defined for this connection.

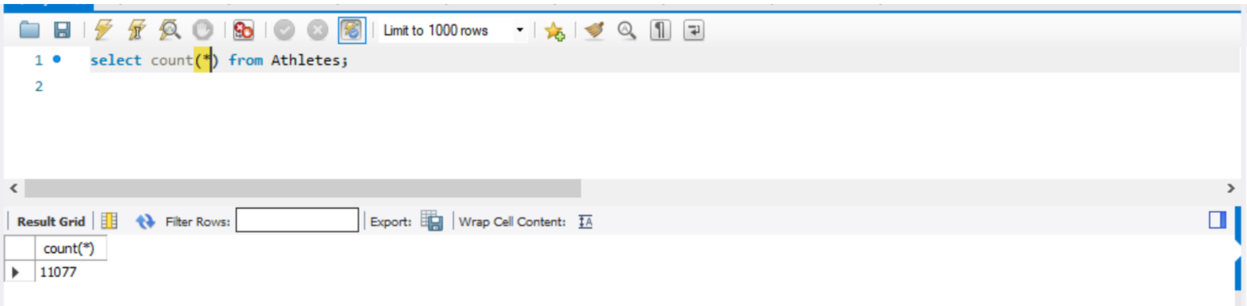
DDL for Table Creation:

```
CREATE TABLE Athletes
(
    Name VARCHAR(512) NOT NULL,
    NOC VARCHAR(512) NOT NULL,
    Discipline  VARCHAR(512) NOT NULL,
    PRIMARY KEY (Name, Discipline)
);
CREATE TABLE Coaches
(
    Name VARCHAR(512),
    NOC VARCHAR(512),
    Discipline  VARCHAR(512),
    Event VARCHAR(512),
    PRIMARY KEY (Name, Event)
);
CREATE TABLE Medals
(
    NOC VARCHAR(512),
    RankOverall INT,
    Gold INT,
    Silver INT,
    Bronze INT,
    Total INT,
    RankByNumberOfMedals INT
    PRIMARY 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),
    PRIMARY KEY (name)
);
```

Number of Entries per table:

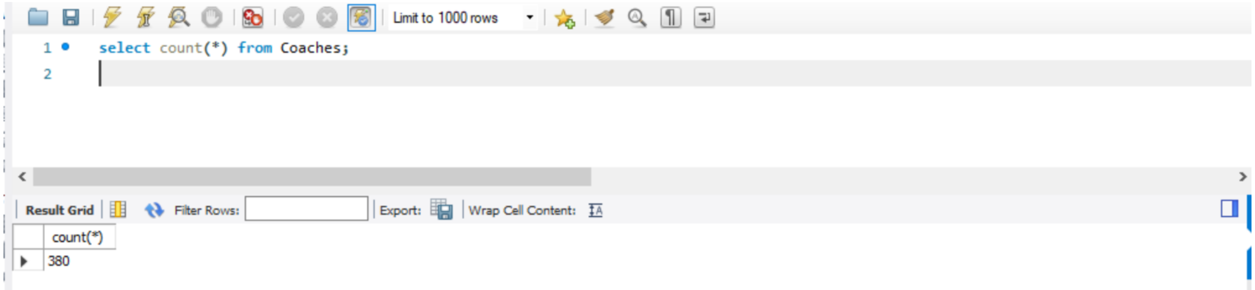
Athletes:



```sql
select count(*) from Athletes;
```

```
count(*)
11077
```

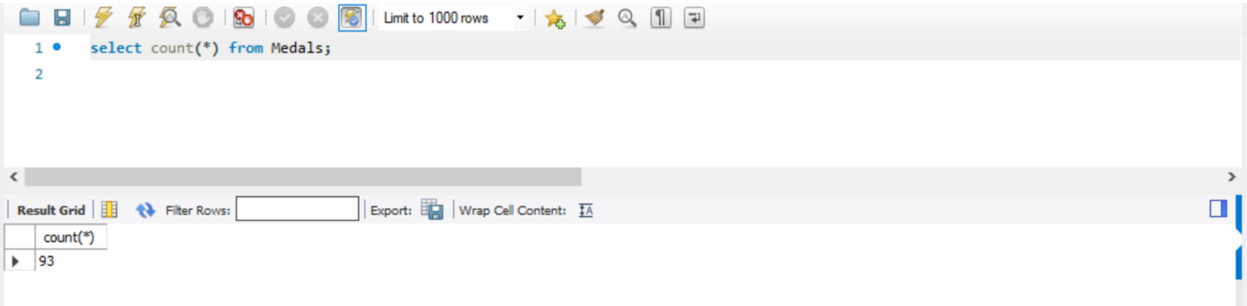
Coaches:



```sql
select count(*) from Coaches;
```

```
count(*)
380
```

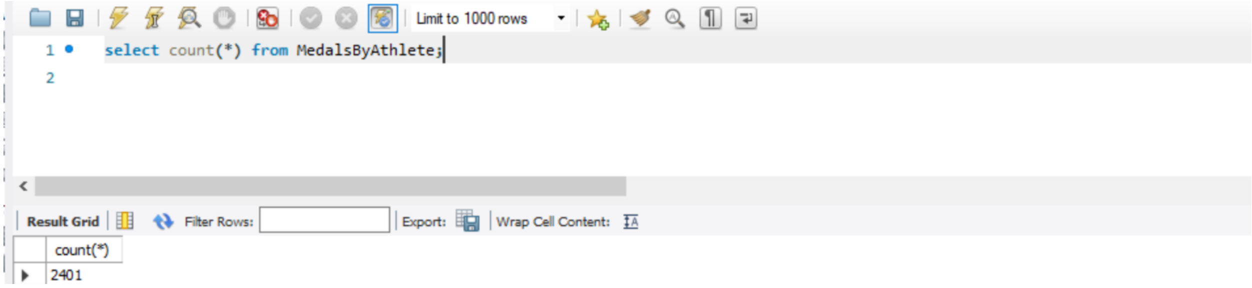
Medals:



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

```
count(*)
93
```

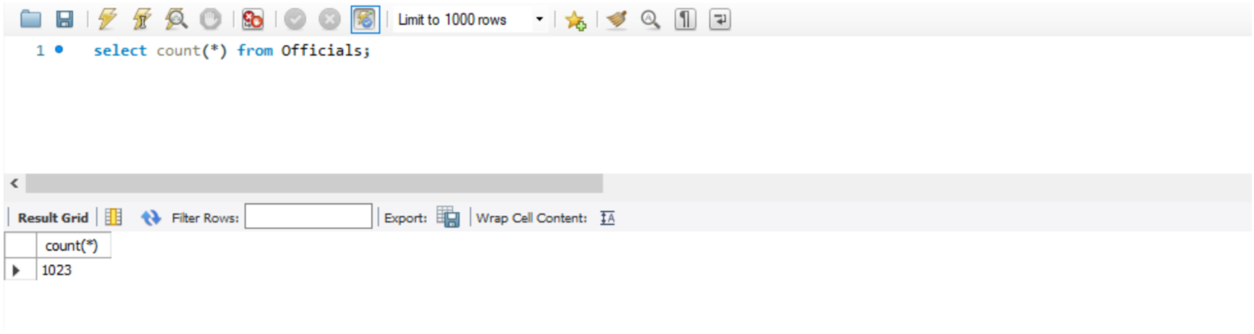
MedalsByAthlete:



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

```
count(*)
2401
```

Officials:



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

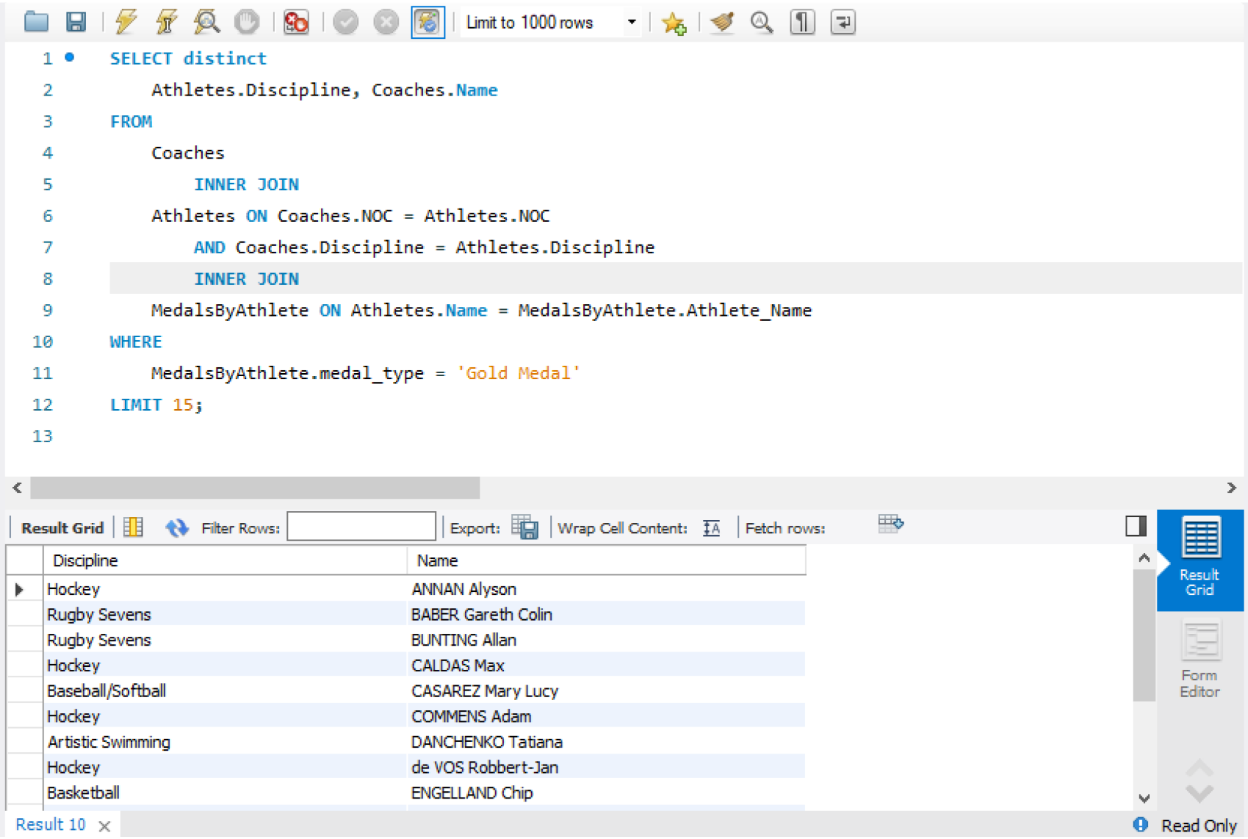
```
count(*)
1023
```

#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'
LIMIT 15;

;
```

Image:



```sql
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'
LIMIT 15;
```

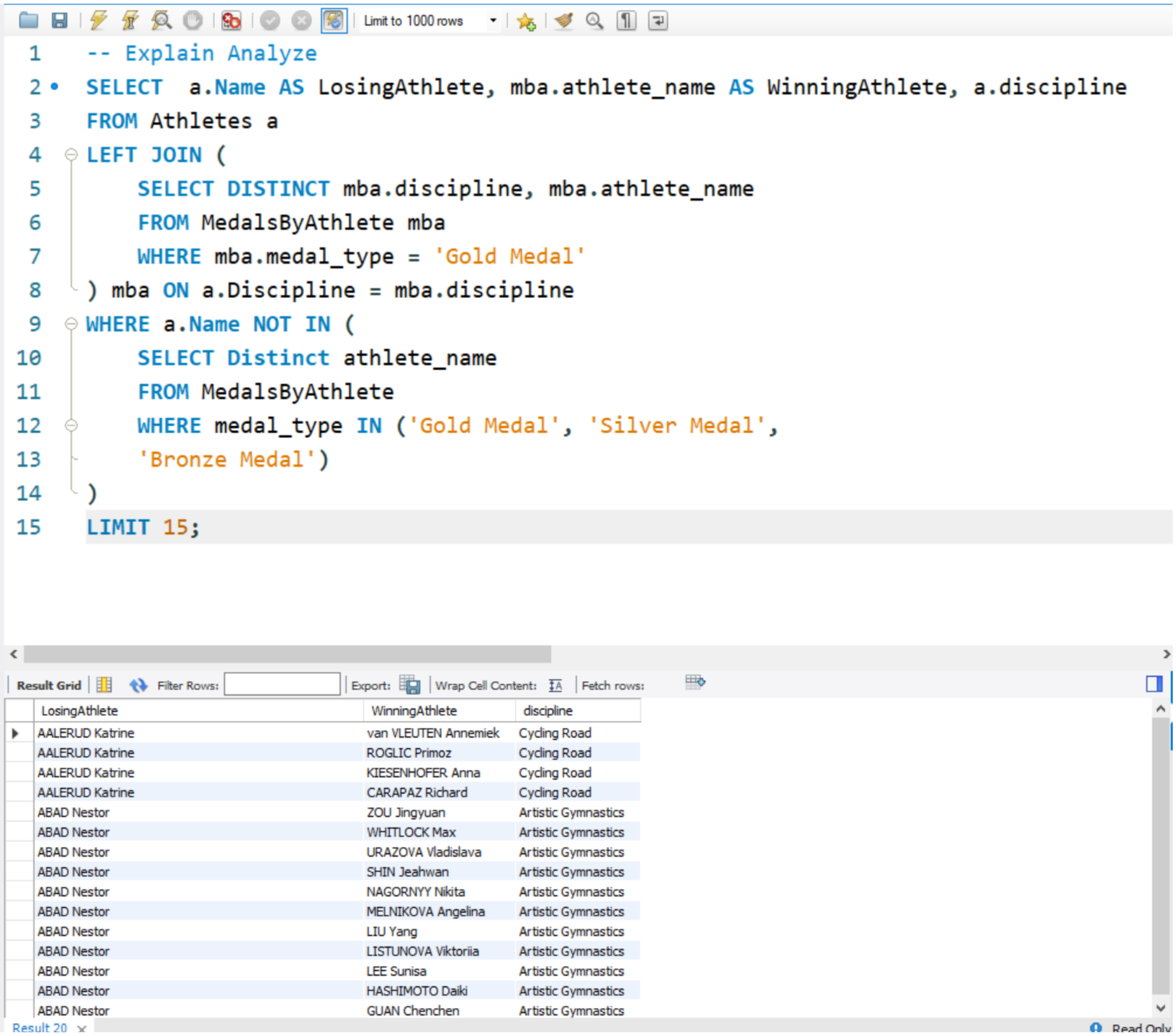
Discipline	Name
Hockey	ANNAN Alyson
Rugby Sevens	BABER Gareth Colin
Rugby Sevens	BUNTING Allan
Hockey	CALDAS Max
Baseball/Softball	CASAREZ Mary Lucy
Hockey	COMMENS Adam
Artistic Swimming	DANCHENKO Tatiana
Hockey	de VOS Robbert-Jan
Basketball	ENGELLAND Chip

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.

#2 Advanced SQL Query

```sql
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:



```
-- 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;
```

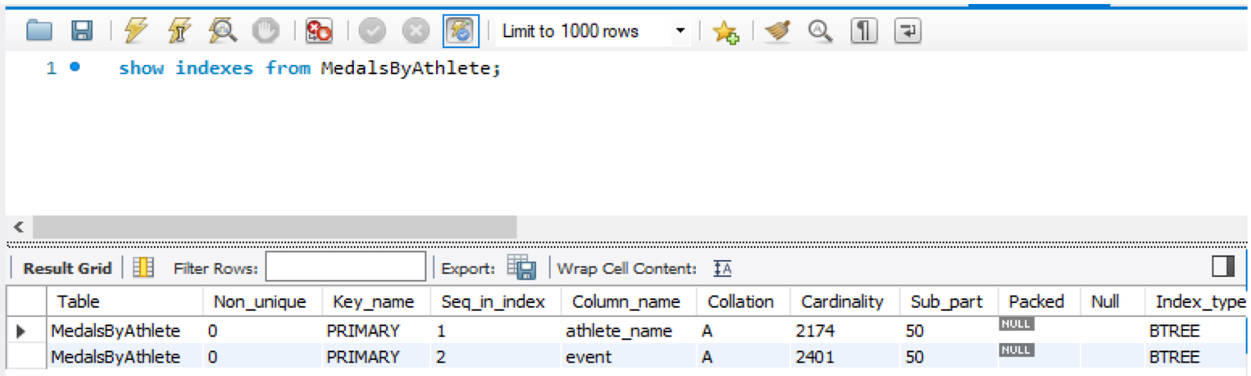
LosingAthlete	WinningAthlete	discipline
AALERUD Katrine	van VLEUTEN Annemiek	Cycling Road
AALERUD Katrine	ROGLIC Primoz	Cycling Road
AALERUD Katrine	KIESENHOFER Anna	Cycling Road
AALERUD Katrine	CARAPAZ Richard	Cycling Road
ABAD Nestor	ZOU Jingyuan	Artistic Gymnastics
ABAD Nestor	WHITLOCK Max	Artistic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

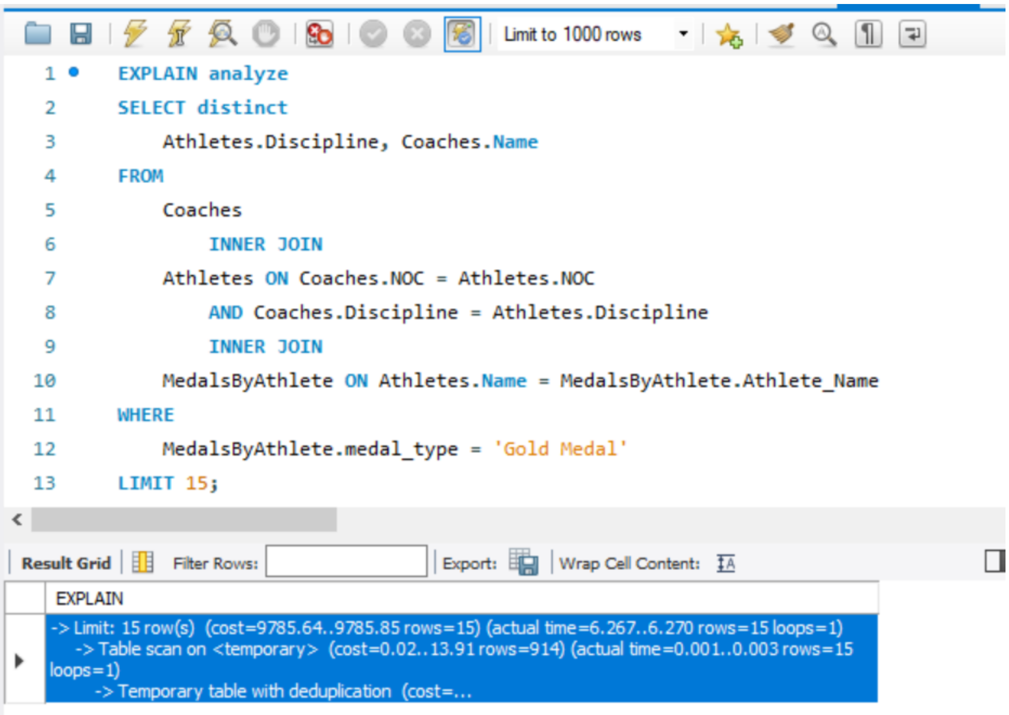
1. Indexes before:



```sql
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

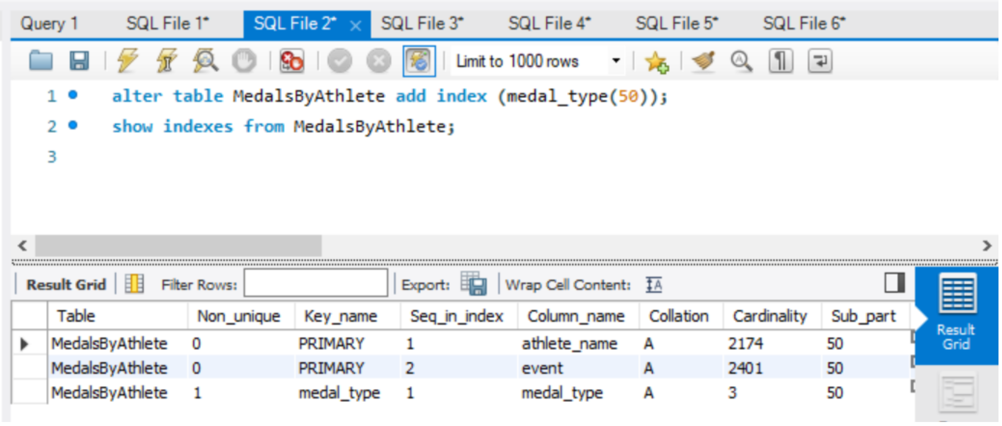
Current runtime:



```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'
LIMIT 15;
```

```
EXPLAIN
-> Limit: 15 row(s)  (cost=9785.64..9785.85 rows=15) (actual time=6.267..6.270 rows=15 loops=1)
    -> Table scan on <temporary>  (cost=0.02..13.91 rows=914) (actual time=0.001..0.003 rows=15
loops=1)
        -> Temporary table with deduplication  (cost=...
```

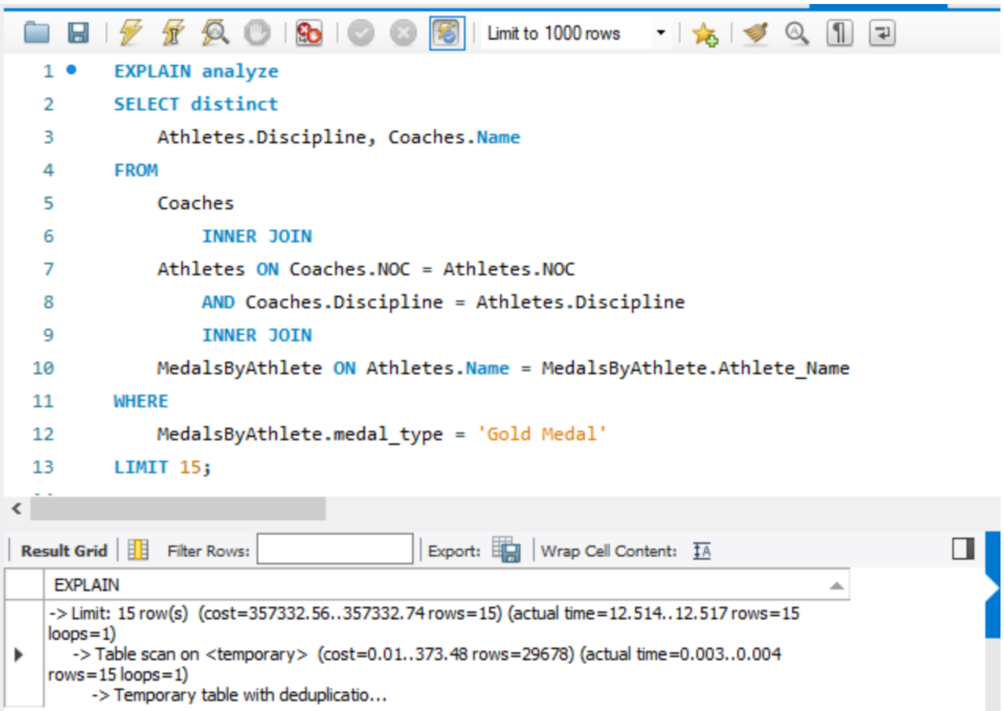
New Indexes:



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

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:

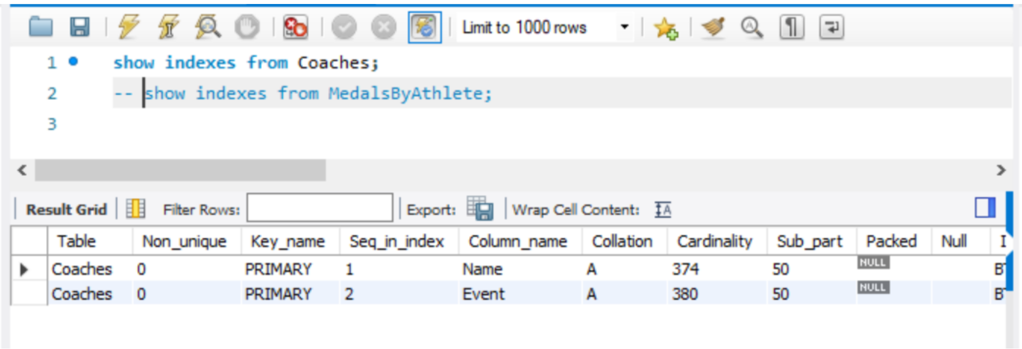


```sql
EXPLAIN analyze
SELECT distinct
    Athletes.Discipline, Coaches.Name
FROM
    Coaches
        INNER JOIN
    Athletes ON Coaches.NOC = Athletes.NOC
        AND Coaches.Discipline = Athletes.Discipline
        INNER JOIN
    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
WHERE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

```
EXPLAIN
-> Limit: 15 row(s)  (cost=357332.56..357332.74 rows=15) (actual time=12.514..12.517 rows=15
loops=1)
    -> Table scan on <temporary>  (cost=0.01..373.48 rows=29678) (actual time=0.003..0.004
rows=15 loops=1)
        -> Temporary table with deduplicatio...
```

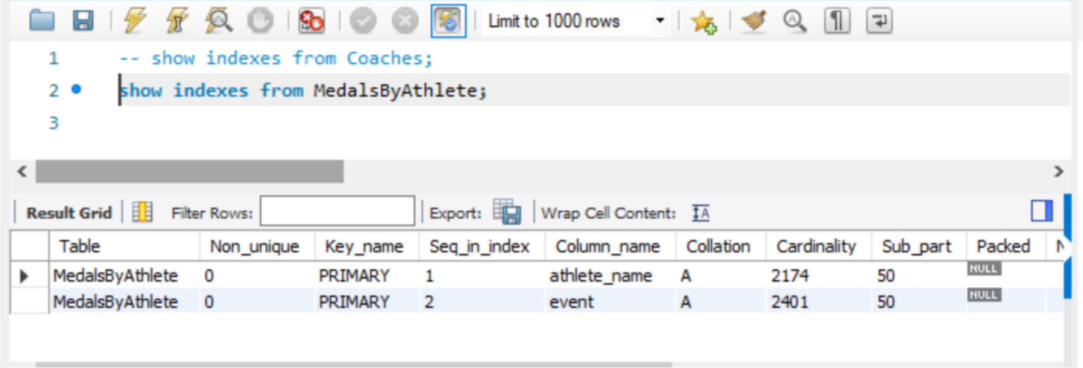
Result: Twice as slow

Default Indexing And Runtime = 6.275:



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

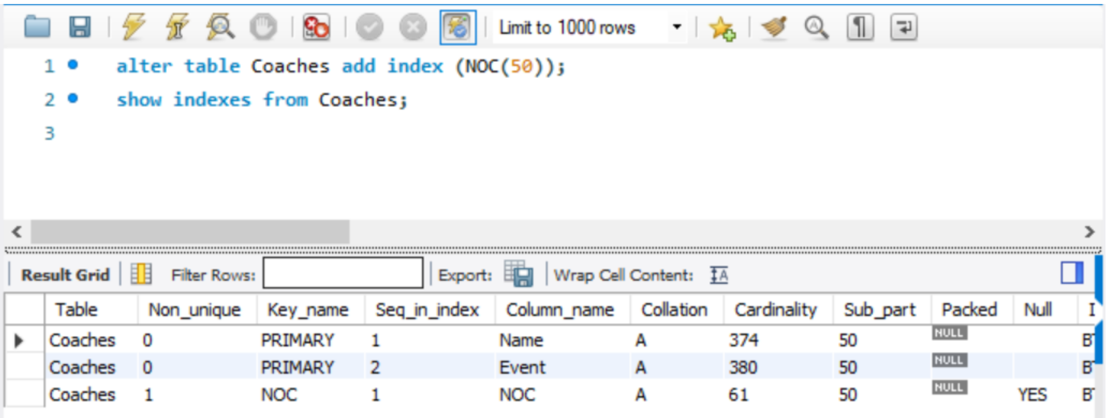
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	I
Coaches	0	PRIMARY	1	Name	A	374	50	NULL		B
Coaches	0	PRIMARY	2	Event	A	380	50	NULL		B



```
-- 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	

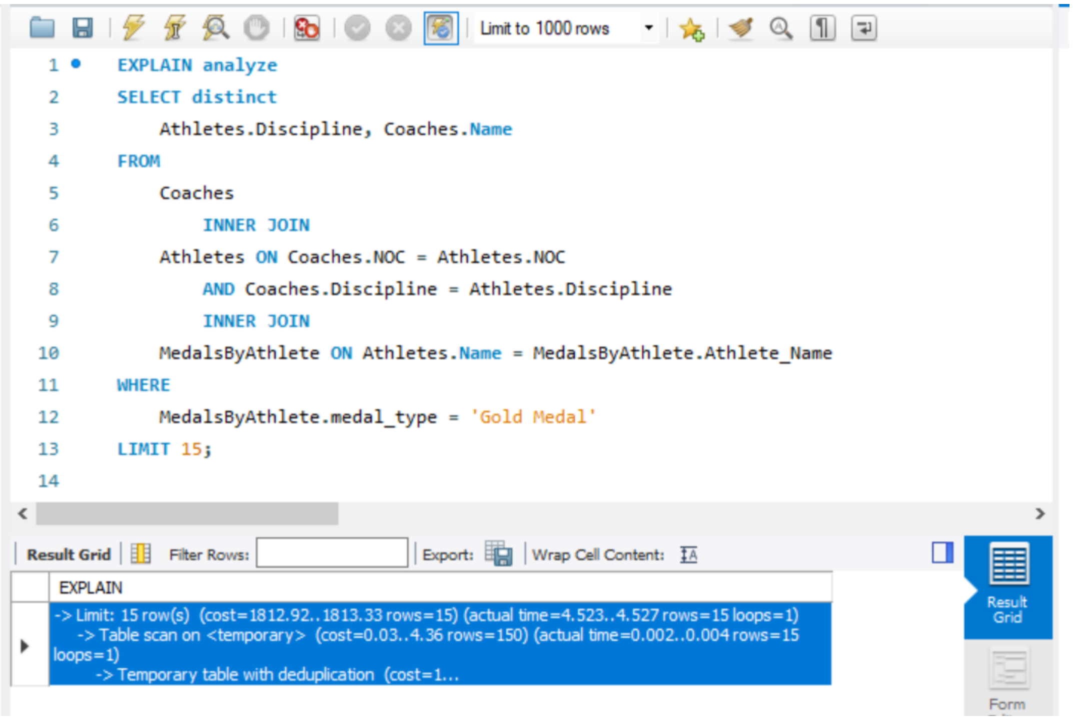
Adding index to Coaches.NOC:



```
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:

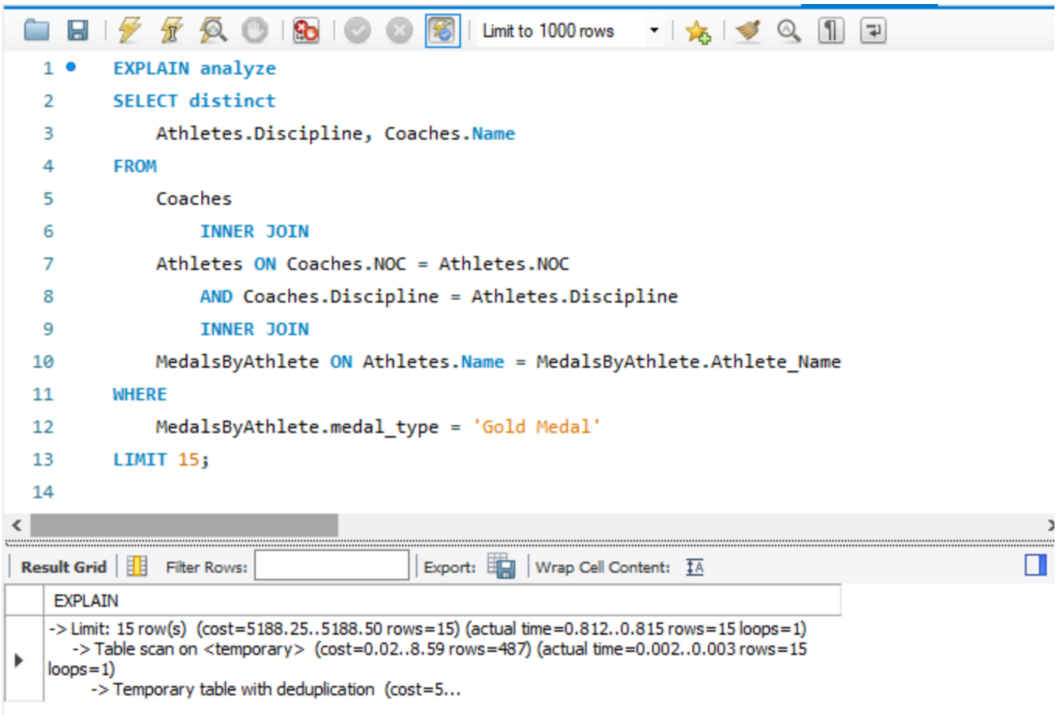


```sql
EXPLAIN analyze
SELECT distinct
    Athletes.Discipline, Coaches.Name
FROM
    Coaches
        INNER JOIN
    Athletes ON Coaches.NOC = Athletes.NOC
        AND Coaches.Discipline = Athletes.Discipline
        INNER JOIN
    MedalsByAthlete ON Athletes.Name = MedalsByAthlete.Athlete_Name
WHERE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

```
EXPLAIN
-> Limit: 15 row(s)  (cost=1812.92..1813.33 rows=15) (actual time=4.523..4.527 rows=15 loops=1)
   -> Table scan on <temporary>  (cost=0.03..4.36 rows=150) (actual time=0.002..0.004 rows=15
loops=1)
      -> Temporary table with deduplication  (cost=1...
```

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:



```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'
LIMIT 15;
```

```
EXPLAIN
-> Limit: 15 row(s)  (cost=5188.25..5188.50 rows=15) (actual time=0.812..0.815 rows=15 loops=1)
    -> Table scan on <temporary>  (cost=0.02..8.59 rows=487) (actual time=0.002..0.003 rows=15
loops=1)
        -> Temporary table with deduplication  (cost=5...
```

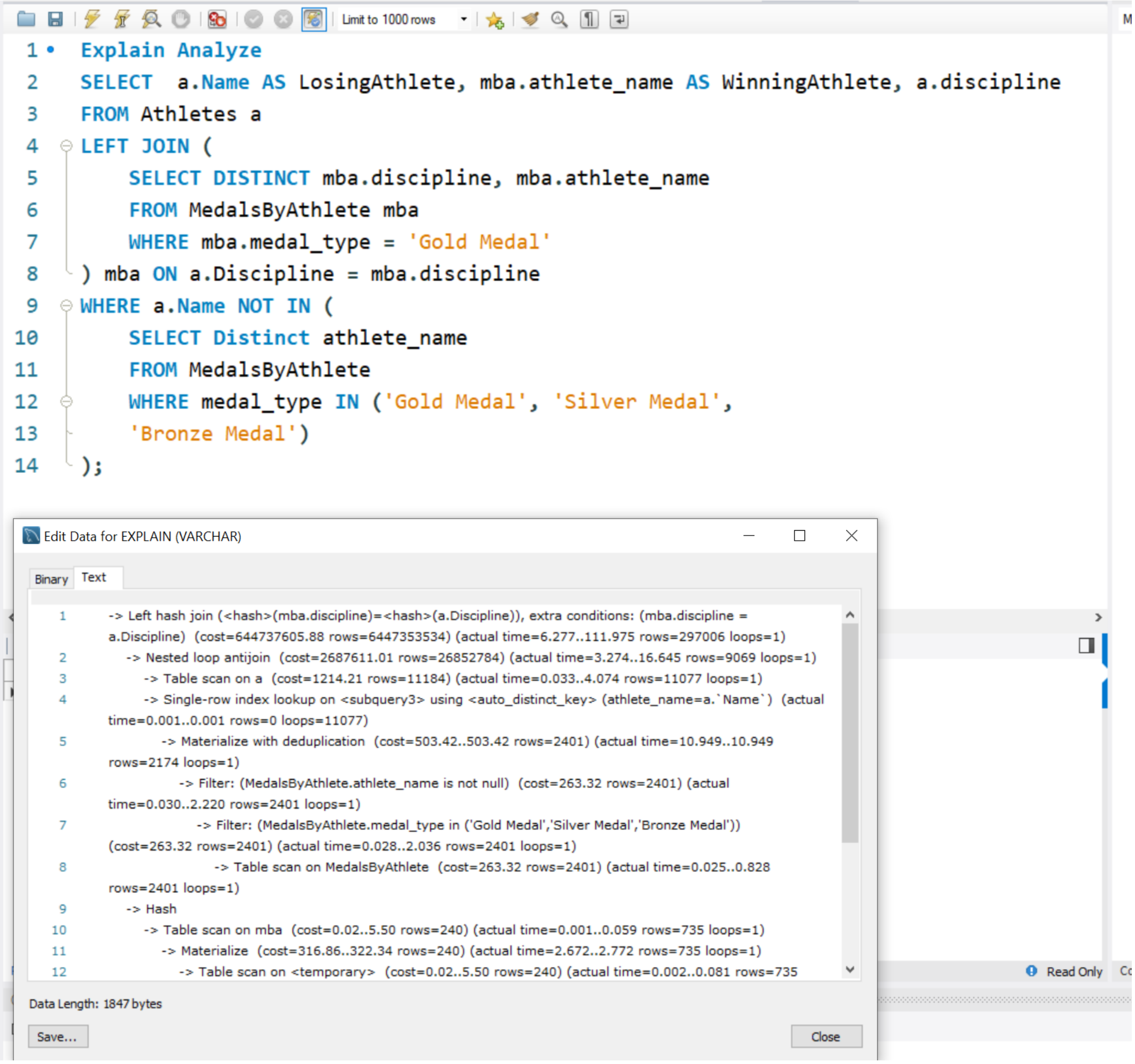
Results: Faster than default indexing, but for some reason adding indexing MedalsByAthlete.medal_type makes the algorithm slower, even when paired with indexing on Coaches.NOC.

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:



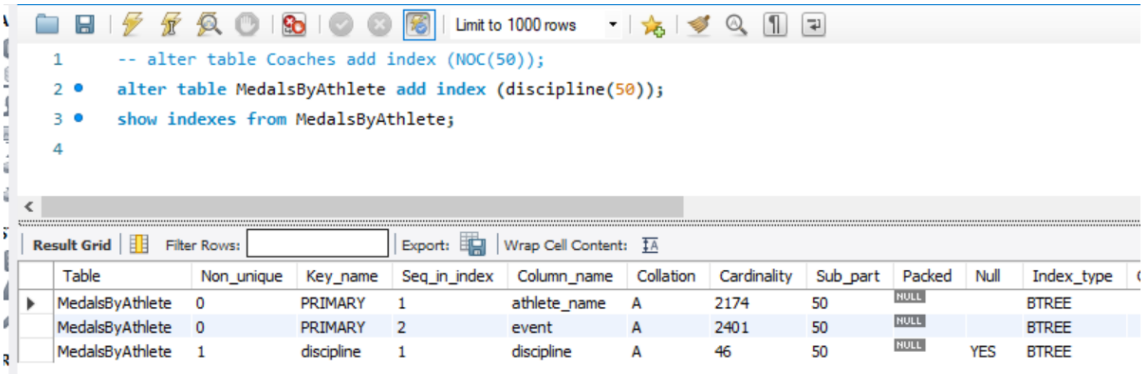
```sql
Explain Analyze
SELECT  a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
FROM Athletes a
LEFT JOIN (
    SELECT DISTINCT mba.discipline, mba.athlete_name
    FROM MedalsByAthlete mba
    WHERE mba.medal_type = 'Gold Medal'
) mba ON a.Discipline = mba.discipline
WHERE a.Name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

Edit Data for EXPLAIN (VARCHAR)

```
-> 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

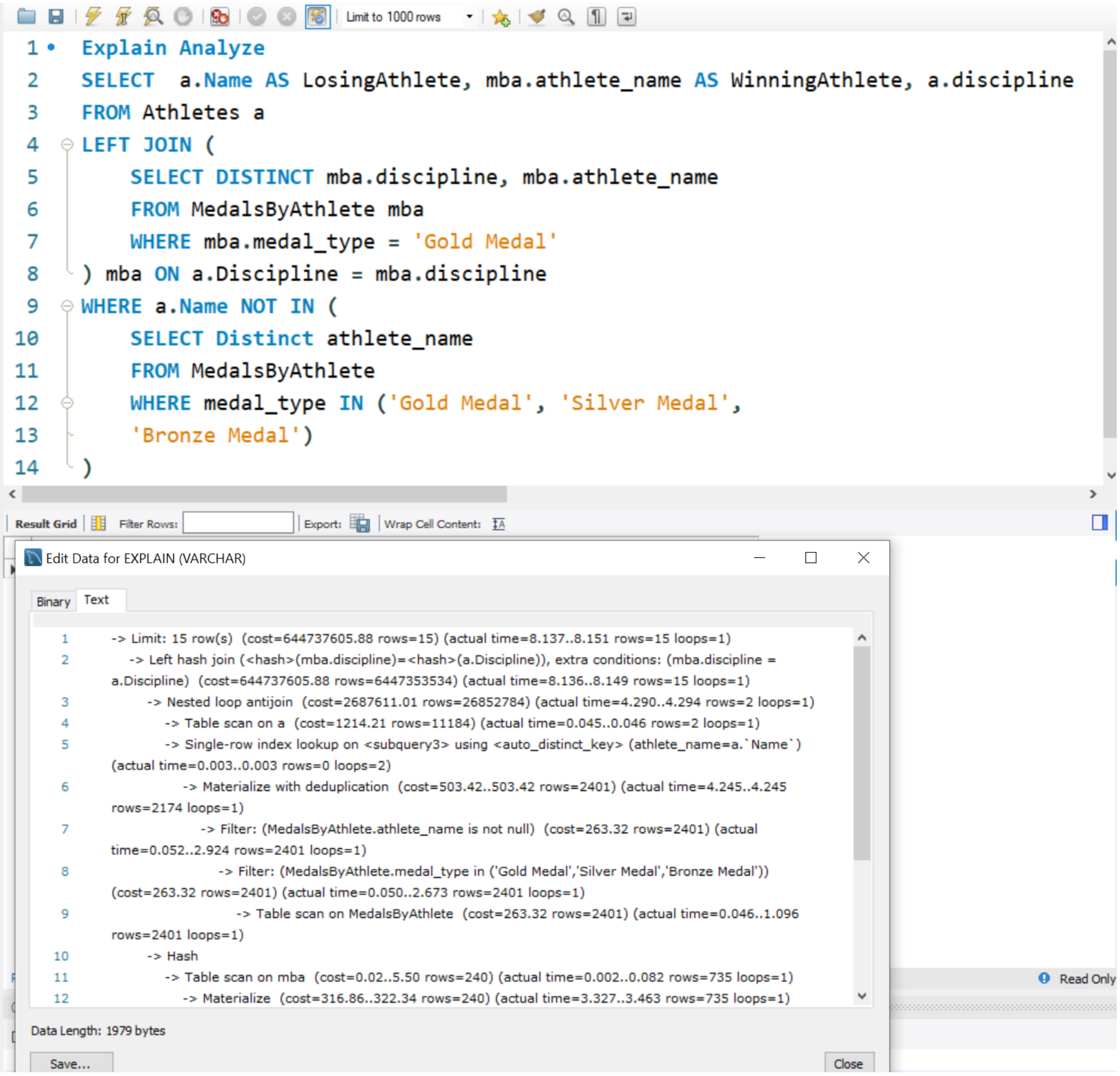
Adding Index on MedalsByAthlete.discipline:



```
-- 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:



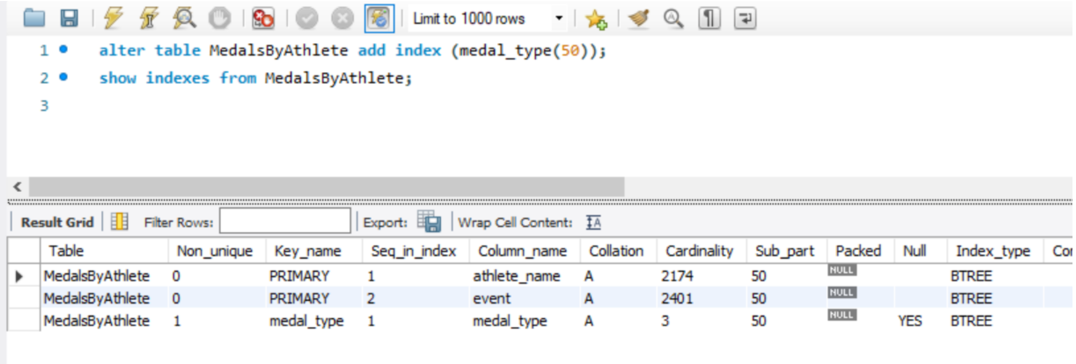
```
Explain Analyze
SELECT  a.Name AS LosingAthlete, mba.athlete_name AS WinningAthlete, a.discipline
FROM Athletes a
LEFT JOIN (
    SELECT DISTINCT mba.discipline, mba.athlete_name
    FROM MedalsByAthlete mba
    WHERE mba.medal_type = 'Gold Medal'
) mba ON a.Discipline = mba.discipline
WHERE a.Name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

```
-> 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

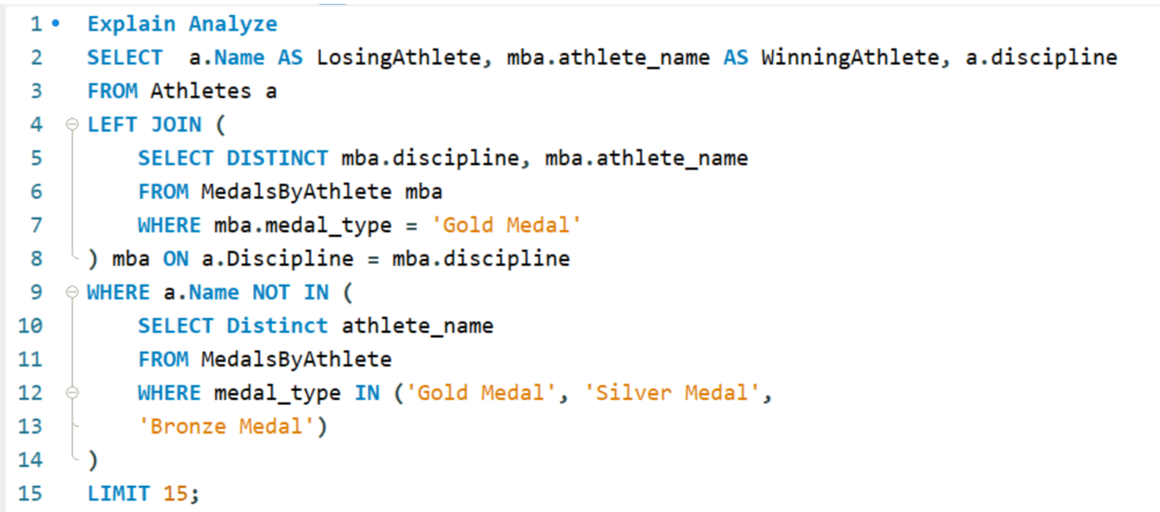
Indexing medal_type instead of discipline:



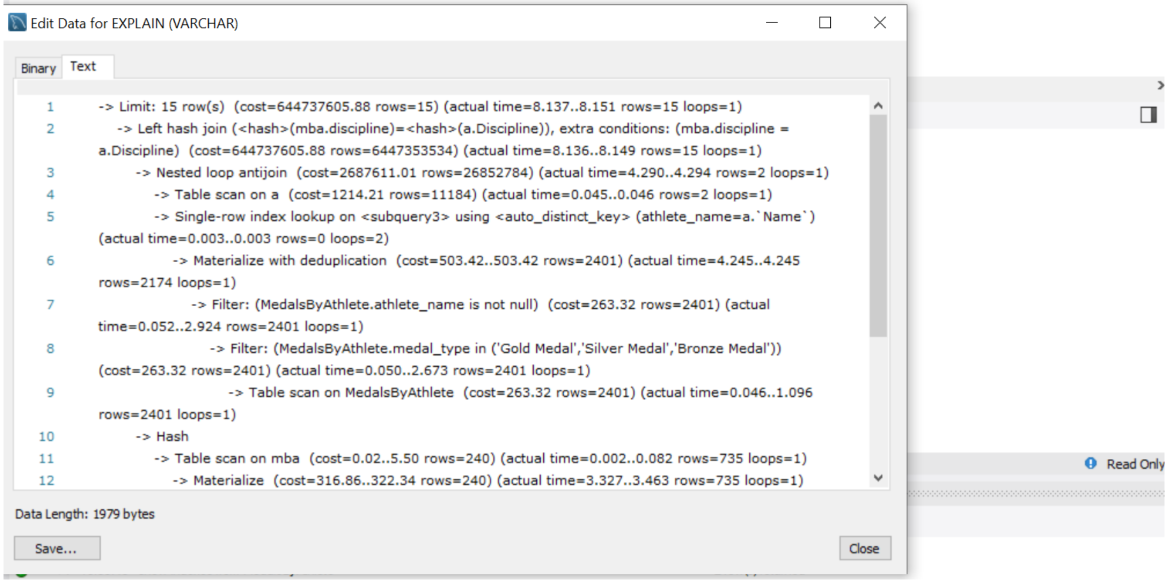
```sql
alter table MedalsByAthlete add index (medal_type(50));
show indexes from MedalsByAthlete;
```

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:



```sql
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)

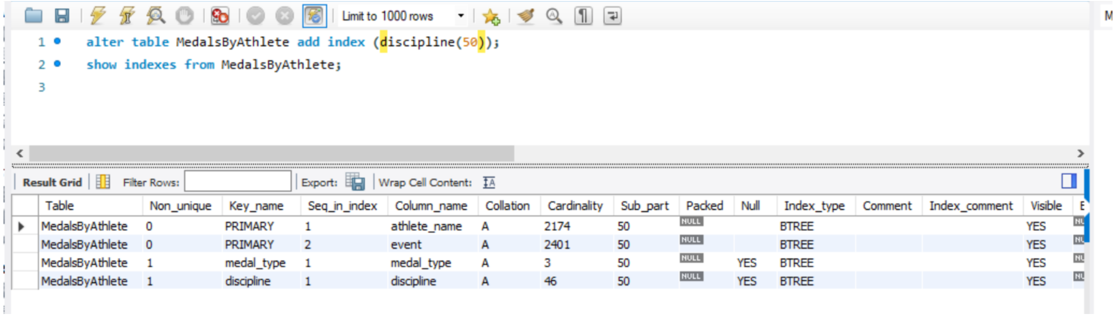
```
-> Limit: 15 row(s)  (cost=644737605.88 rows=15) (actual time=8.137..8.151 rows=15 loops=1)
    -> Left hash join (<hash>(mba.discipline)=<hash>(a.Discipline)), extra conditions: (mba.discipline = a.Discipline)  (cost=644737605.88 rows=6447353534) (actual time=8.136..8.149 rows=15 loops=1)
        -> 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:

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

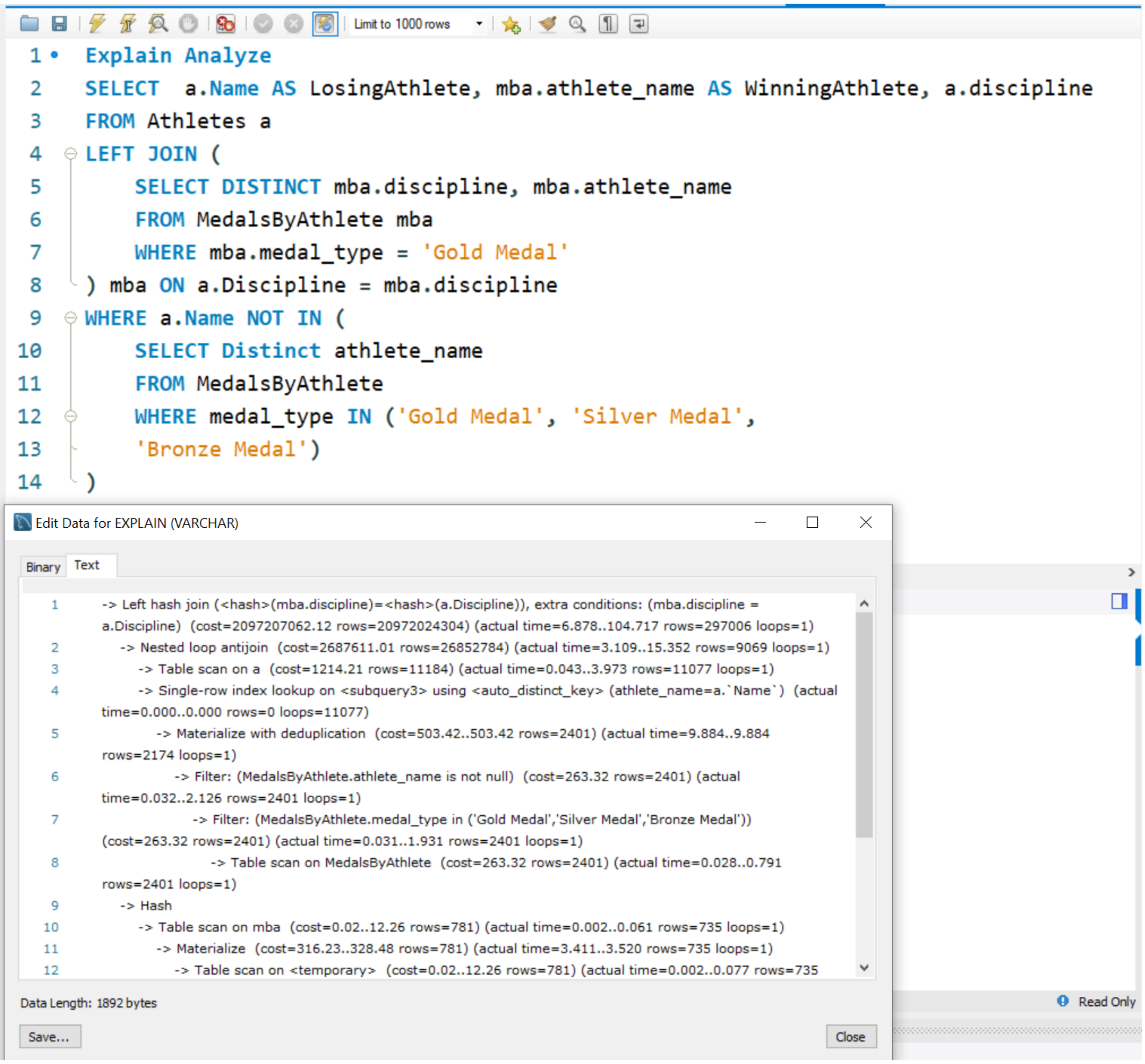
Indexing both discipline and medal_type:



```sql
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:



```sql
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')
)
```

```
-> 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

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.