

## DATABASE IMPLEMENTATION TABLES

### Created table for Judges

```
queries.sql 3ykja6tyw NEW MYSQL RUN

1 CREATE TABLE judge(
2   judgeName VARCHAR(255) NOT NULL
3   ,nationality VARCHAR(3) NOT NULL
4   ,role VARCHAR(2) NOT NULL
5   ,segmentCategory VARCHAR(255) NOT NULL
6   ,program VARCHAR(255) NOT NULL
7   ,competition VARCHAR(255) NOT NULL
8   ,PRIMARY KEY(judgeName,role,program,competition)
9 );
10 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Chihee RHEE
11 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. David MUNOZ
12 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Feng HUANG'
13 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Richard DAL
14 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Walter ZUCC
15 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Leanna CARO
16 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Kaoru TAKIN
17 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Christine H
18 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Elena FOMIN
19 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Walter ZUCC
20 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Feng HUANG'
21 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Leanna CARO
22 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Elena FOMIN
23 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Richard DAL
24 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Chihee RHEE
25 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Kaoru TAKIN
26 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. David MUNOZ
27 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Christine H
28 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Paolo RIZZO
29 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Saodat NUPA
30 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Olga KOZHEM
31 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Kenji AMAKO
32 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Elizabeth A
33 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Chihee RHEE
34 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Tamie K. CA
35 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Weiguang CH
36 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Jeff LUKASI
37 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Anthony LER
38 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Olga KOZHEM
39 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Mr. Kersten BEL
40 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Leanna CARO
41 INSERT INTO judge(judgeName,nationality,role,segmentCategory,program,competition) VALUES ('Ms. Sakae YAMAM
42
```

STDIN  
Input for the program ( Optional )

Output:  
Count(judgeName)  
1368

### Created table for Skaters

```
1 CREATE TABLE skater(
2   name VARCHAR(39) NOT NULL PRIMARY KEY
3   ,nationality VARCHAR(3) NOT NULL
4 );
5
6 INSERT IGNORE INTO skater(name,nationality) VALUES ('Aliona SAVCHENKO / Bruno MASSOT','GER');
7 INSERT IGNORE INTO skater(name,nationality) VALUES ('Evgenia TARASOVA / Vladimir MOROZOV','RUS');
8 INSERT IGNORE INTO skater(name,nationality) VALUES ('Natalia ZABIIAKO / Alexander ENBERT','RUS');
9 INSERT IGNORE INTO skater(name,nationality) VALUES ('Vanessa JAMES / Morgan CIPRES','FRA');
10 INSERT IGNORE INTO skater(name,nationality) VALUES ('Marissa CASTELLI / Mervin TRAN','USA');
11 INSERT IGNORE INTO skater(name,nationality) VALUES ('Miriam ZIEGLER / Severin KIEFER','AUT');
12 INSERT IGNORE INTO skater(name,nationality) VALUES ('Aliona SAVCHENKO / Bruno MASSOT','GER');
13 INSERT IGNORE INTO skater(name,nationality) VALUES ('Vanessa JAMES / Morgan CIPRES','FRA');
14 INSERT IGNORE INTO skater(name,nationality) VALUES ('Evgenia TARASOVA / Vladimir MOROZOV','RUS');
15 INSERT IGNORE INTO skater(name,nationality) VALUES ('Natalia ZABIIAKO / Alexander ENBERT','RUS');
16 INSERT IGNORE INTO skater(name,nationality) VALUES ('Marissa CASTELLI / Mervin TRAN','USA');
17 INSERT IGNORE INTO skater(name,nationality) VALUES ('Miriam ZIEGLER / Severin KIEFER','AUT');
18 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabriella PAPADAKIS / Guillaume CIZERON','FRA');
19 INSERT IGNORE INTO skater(name,nationality) VALUES ('Elena ILINYKH / Ruslan ZHIGANSHIN','RUS');
20 INSERT IGNORE INTO skater(name,nationality) VALUES ('Madison HUBBELL / Zachary DONOHUE','USA');
21 INSERT IGNORE INTO skater(name,nationality) VALUES ('Piper GILLES / Paul POIRIER','CAN');
22 INSERT IGNORE INTO skater(name,nationality) VALUES ('Isabella TOBIAS / Ilia TKACHENKO','ISR');
23 INSERT IGNORE INTO skater(name,nationality) VALUES ('Marie-Jade LAURIAULT / Romain LE GAC','FRA');
24 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alexandra NAZAROVA / Maxim NIKITIN','UKR');
25 INSERT IGNORE INTO skater(name,nationality) VALUES ('Cortney MANSOUR / Michal CESKA','CZE');
26 INSERT IGNORE INTO skater(name,nationality) VALUES ('Lorenza ALESSANDRINI / Pierre SOUQUET','FRA');
27 INSERT IGNORE INTO skater(name,nationality) VALUES ('Viktoria KAVALIOVA / Yurii BIELIAIEV','BLR');
28 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabriella PAPADAKIS / Guillaume CIZERON','FRA');
```

## Created table for Scores

```
queries.sql 3ykja6tyw NEW MYSQL RUN

1 CREATE TABLE score(
2   skaterName VARCHAR(255) NOT NULL
3   ,competitionName VARCHAR(255) NOT NULL
4   ,program VARCHAR(255) NOT NULL
5   ,judgeRole VARCHAR(2) NOT NULL
6   ,avgComponentScore FLOAT NOT NULL
7   ,PRIMARY KEY(skaterName,competitionName,program,judgeRole)
8 );
9 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
10 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
11 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
12 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
13 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
14 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
15 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
16 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
17 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Aliona SAVCHENK
18 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
19 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
20 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
21 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
22 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
23 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
24 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
25 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
26 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Evgenia TARASOV
27 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
28 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
29 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
30 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
31 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
32 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
33 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
34 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
35 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Natalia ZABIIAK
36 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Vanessa JAMES /
37 INSERT INTO score(skaterName,competitionName,program,judgeRole,avgComponentScore) VALUES ('Vanessa JAMES /
38
```

STDIN

Input for the program ( Optional )

Output:

```
COUNT(*)
1422
```

## Created table for Competitions

```
queries.sql 3ykbhbfjm

1 CREATE TABLE competition(
2   name VARCHAR(255) NOT NULL PRIMARY KEY
3   -- location, start date, end date not included for now
4 );
5
6 CREATE TABLE admin(
7   userID VARCHAR(255) NOT NULL PRIMARY KEY,
8   password VARCHAR(255) NOT NULL
9 );
10
11
12 INSERT INTO competition(name) VALUES ('ISU European Figure Skating Championships 2017');
13 INSERT INTO competition(name) VALUES ('ISU Four Continents Championships 2017');
14 INSERT INTO competition(name) VALUES ('ISU GP 2016 Skate Canada International');
15 INSERT INTO competition(name) VALUES ('ISU GP 2017 Skate Canada International');
16 INSERT INTO competition(name) VALUES ('ISU GP Audi Cup of China 2016');
17 INSERT INTO competition(name) VALUES ('ISU GP Audi Cup of China 2017');
18 INSERT INTO competition(name) VALUES ('ISU Grand Prix of Figure Skating Final 2016');
19 INSERT INTO competition(name) VALUES ('Grand Prix Final 2017 Senior and Junior');
20 INSERT INTO competition(name) VALUES ('ISU GP Trophée de France 2016');
21 INSERT INTO competition(name) VALUES ('ISU GP Internationaux de France de Patinage 2017');
22 INSERT INTO competition(name) VALUES ('ISU GP NHK Trophy 2016');
23 INSERT INTO competition(name) VALUES ('ISU GP NHK Trophy 2017');
24 INSERT INTO competition(name) VALUES ('ISU GP Rostelecom Cup 2016');
```

## Created table for Admins

```
1 CREATE TABLE Admin(
2   userID VARCHAR(255) NOT NULL PRIMARY KEY
3   ,password VARCHAR(255) NOT NULL
4 );
```

## TWO ADVANCED SQL QUERIES

Average score judge gives to skaters of different nationalities

queries.sql3ykja6tywNEWMySQLRUN

```
858 INSERT IGNORE INTO skater(name,nationality) VALUES ('Evgenia MEDVEDEVA','RUS');
859 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabrielle DALEMAN','CAN');
860 INSERT IGNORE INTO skater(name,nationality) VALUES ('Maria SOTSKOVA','RUS');
861 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine LECAVELIER','FRA');
862 INSERT IGNORE INTO skater(name,nationality) VALUES ('Wakaba HIGUCHI','JPN');
863 INSERT IGNORE INTO skater(name,nationality) VALUES ('So Youn PARK','KOR');
864 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena LEONOVA','RUS');
865 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao ASADA','JPN');
866 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia GALUSTYAN','ARM');
867 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie GOLD','USA');
868 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mae Berenice METTE','FRA');
869 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka NAGAI','JPN');
870 INSERT IGNORE INTO skater(name,nationality) VALUES ('Evgenia MEDVEDEVA','RUS');
871 INSERT IGNORE INTO skater(name,nationality) VALUES ('Maria SOTSKOVA','RUS');
872 INSERT IGNORE INTO skater(name,nationality) VALUES ('Wakaba HIGUCHI','JPN');
873 INSERT IGNORE INTO skater(name,nationality) VALUES ('So Youn PARK','KOR');
874 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mae Berenice METTE','FRA');
875 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabrielle DALEMAN','CAN');
876 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine LECAVELIER','FRA');
877 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie GOLD','USA');
878 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka NAGAI','JPN');
879 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao ASADA','JPN');
880 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia GALUSTYAN','ARM');
881 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena LEONOVA','RUS');
882
883 -- average score judge gives skaters of different nationalities
884
885 SELECT judgeName, judge.nationality AS judgeNat, skater.nationality AS skaterNat, AVG(avgComponentsScore)
886 FROM skater JOIN score ON name = skaterName JOIN judge ON competitionName = competition AND score.program
887 GROUP BY judgeName, judge.nationality, skater.nationality
888 LIMIT 15;
```

STDIN  
Input for the program (Optional)

Output:

judgeName	judgeNat	skaterNat	avgScoreFromJudge
Mr. Alexandre GOROJDANOV	BLR	BLR	6.900000
Mr. Alexandre GOROJDANOV	BLR	CAN	8.750000
Mr. Alexandre GOROJDANOV	BLR	FRA	7.600000
Mr. Alexandre GOROJDANOV	BLR	USA	8.600000
Mr. Alexandre GOROJDANOV	BLR	ISR	8.575000
Mr. Alexandre GOROJDANOV	BLR	RUS	9.125000
Mr. Alexandre GOROJDANOV	BLR	CZE	6.625000
Mr. Alexandre GOROJDANOV	BLR	UKR	7.625000
Mr. Alexei BELETSKI	ISR	BLR	6.375000
Mr. Alexei BELETSKI	ISR	CAN	8.500000
Mr. Alexei BELETSKI	ISR	FRA	7.708333
Mr. Alexei BELETSKI	ISR	USA	8.475000
Mr. Alexei BELETSKI	ISR	ISR	8.550000
Mr. Alexei BELETSKI	ISR	RUS	8.675000
Mr. Alexei BELETSKI	ISR	CZE	6.975000

Select the highest and lowest average score a judge gives to a nationality and compare them to average score they give per nationality

queries.sql3ykja6tywNEWMySQLRUN

```
859 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabrielle DALEMAN','CAN');
860 INSERT IGNORE INTO skater(name,nationality) VALUES ('Maria SOTSKOVA','RUS');
861 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine LECAVELIER','FRA');
862 INSERT IGNORE INTO skater(name,nationality) VALUES ('Wakaba HIGUCHI','JPN');
863 INSERT IGNORE INTO skater(name,nationality) VALUES ('So Youn PARK','KOR');
864 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena LEONOVA','RUS');
865 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao ASADA','JPN');
866 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia GALUSTYAN','ARM');
867 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie GOLD','USA');
868 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mae Berenice METTE','FRA');
869 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka NAGAI','JPN');
870 INSERT IGNORE INTO skater(name,nationality) VALUES ('Evgenia MEDVEDEVA','RUS');
871 INSERT IGNORE INTO skater(name,nationality) VALUES ('Maria SOTSKOVA','RUS');
872 INSERT IGNORE INTO skater(name,nationality) VALUES ('Wakaba HIGUCHI','JPN');
873 INSERT IGNORE INTO skater(name,nationality) VALUES ('So Youn PARK','KOR');
874 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mae Berenice METTE','FRA');
875 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabrielle DALEMAN','CAN');
876 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine LECAVELIER','FRA');
877 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie GOLD','USA');
878 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka NAGAI','JPN');
879 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao ASADA','JPN');
880 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia GALUSTYAN','ARM');
881 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena LEONOVA','RUS');
882
883 -- average score judge gives skaters of different nationalities
884
885 -- SELECT judgeName, judge.nationality AS judgeNat, skater.nationality AS skaterNat, AVG(avgComponentsScore)
886 -- FROM skater JOIN score ON name = skaterName JOIN judge ON competitionName = competition AND score.program
887 -- GROUP BY judgeName, judge.nationality, skater.nationality
888 -- LIMIT 15;
889
890 -- select the highest and lowest avg score a judge gives to a nationality and compare to average score they give per nationality
891
892 SELECT judgeName, MAX(avgScoreFromJudge), MIN(avgScoreFromJudge), AVG(avgScoreFromJudge)
893 FROM (
894 SELECT judgeName, judge.nationality AS judgeNat, skater.nationality AS skaterNat, AVG(avgComponentsScore)
895 FROM skater JOIN score ON name = skaterName JOIN judge ON competitionName = competition AND score.program
896 GROUP BY judgeName, judge.nationality, skater.nationality) avg
897 GROUP BY judgeName
898 LIMIT 15;
```

STDIN  
Input for the program (Optional)

Output:

judgeName	MAX(avgScoreFromJudge)	MIN(avgScoreFromJudge)	AVG(avgScoreFromJudge)
Mr. Alexandre GOROJDANOV	9.125000	6.625000	7.975000000000
Mr. Alexei BELETSKI	8.675000	6.375000	7.851666250
Mr. Daniel DELFA	9.325000	5.850000	7.6829860833
Mr. John MILLER	9.150000	5.800000	7.6135416250
Mr. Massimo ORLANDINI	9.175000	6.050000	7.8142360833
Mr. Richard KOSINA	8.287500	6.250000	7.4372727273
Mr. Steve WINKLER	9.150000	5.875000	7.6770833333
Mr. Yuri GUSKOV	9.200000	6.500000	7.9447222000
Ms. Christiane MILLER	9.150000	6.525000	7.5657407778
Ms. Christine HURTH	8.750000	6.100000	7.6635416250
Ms. Dagmar LURZ-PROTT	8.100000	6.550000	7.4571428571
Ms. Elena LISOVA	9.200000	6.625000	7.9885416250
Ms. Elisabeth LOUESDON	9.125000	6.575000	7.6337500000
Ms. Francoise DE RAPPARD	9.125000	6.850000	7.7037500000
Ms. Helene CUCUPHAT	8.850000	6.025000	7.8050000000

## INDEXING ANALYSIS

### Query #1

Performance without indexes:

Schema SQL

```
576 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine
LECAVELIER','FRA');
877 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie
GOLD','USA');
878 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka
NAGAI','JPN');
879 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao
ASADA','JPN');
880 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia
GALUSTYAN','ARM');
881 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena
LEONOVA','RUS');
882
883 # CREATE INDEX idx_nat_skater ON skater(nationality);
```

Text to DDL

Query SQL

```
1 -- average score judge gives skaters of different nationalities
2
3 EXPLAIN ANALYZE SELECT judgeName, judge.nationality AS judgeNat,
skater.nationality AS skaterNat, AVG(avgComponentScore) AS
avgScoreFromJudge
4 FROM skater JOIN score ON name = skaterName JOIN judge ON
competitionName = competition AND score.program = judge.program
AND judgeRole = role
5 GROUP BY judgeName, judge.nationality, skater.nationality;
6
7 -- select the highest and lowest avg score a judge gives to a
nationality and compare to average score they give per
nationality
```

Copy as Markdown

Results

Query #1 Execution time: 32ms

EXPLAIN

-> Table scan on <temporary> (actual time=31.452..31.495 rows=226 loops=1) -> Aggregate using temporary table (actual time=31.451..31.451 rows=226 loops=1) -> Nested loop inner join (cost=1268.01 rows=2808) (actual time=0.208..30.368 rows=702 loops=1) -> Inner hash join (no condition) (cost=285.21 rows=2808) (actual time=0.149..0.513 rows=2808 loops=1) -> Table scan on judge (cost=0.19 rows=72) (actual time=0.090..0.141 rows=72 loops=1) -> Hash -> Table scan on skater (cost=4.15 rows=39) (actual time=0.023..0.041 rows=39 loops=1) -> Filter: (score.program = judge.program) (cost=0.25 rows=1) (actual time=0.010..0.011 rows=0 loops=2808) -> Single-row index lookup on score using PRIMARY (skaterName=skater.name, competitionName=judge.competition, program=judge.program, judgeRole=judge.role) (cost=0.25 rows=1) (actual time=0.010..0.010 rows=0 loops=2808)

### Query #1a

Performance with index on skater nationality. This query was chosen because it is the one of the attributes that is used in the aggregate group-by function. Thus, we thought that being able to locate skaters of the same nationalities with a clustered index would benefit the runtime of the aggregation. We can see that the runtime of the query decreased from 32ms to 24ms.

Schema SQL

```
576 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine
LECAVELIER','FRA');
877 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie
GOLD','USA');
878 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka
NAGAI','JPN');
879 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao
ASADA','JPN');
880 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia
GALUSTYAN','ARM');
881 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena
LEONOVA','RUS');
882
883 CREATE INDEX idx_nat_skater ON skater(nationality);
```

Text to DDL

Query SQL

```
1 -- average score judge gives skaters of different nationalities
2
3 EXPLAIN ANALYZE SELECT judgeName, judge.nationality AS judgeNat,
skater.nationality AS skaterNat, AVG(avgComponentScore) AS
avgScoreFromJudge
4 FROM skater JOIN score ON name = skaterName JOIN judge ON
competitionName = competition AND score.program = judge.program
AND judgeRole = role
5 GROUP BY judgeName, judge.nationality, skater.nationality;
6
7 -- select the highest and lowest avg score a judge gives to a
nationality and compare to average score they give per
nationality
```

Copy as Markdown

Results

Query #1 Execution time: 24ms

EXPLAIN

-> Table scan on <temporary> (actual time=23.166..23.203 rows=226 loops=1) -> Aggregate using temporary table (actual time=23.164..23.164 rows=226 loops=1) -> Nested loop inner join (cost=1268.01 rows=2808) (actual time=0.129..22.167 rows=702 loops=1) -> Inner hash join (no condition) (cost=285.21 rows=2808) (actual time=0.059..0.404 rows=2808 loops=1) -> Table scan on judge (cost=0.19 rows=72) (actual time=0.025..0.066 rows=72 loops=1) -> Hash -> Covering index scan on skater using idx\_nat\_skater (cost=4.15 rows=39) (actual time=0.016..0.023 rows=39 loops=1) -> Filter: (score.program = judge.program) (cost=0.25 rows=1) (actual time=0.008..0.008 rows=0 loops=2808) -> Single-row index lookup on score using PRIMARY (skaterName=skater.name, competitionName=judge.competition, program=judge.program, judgeRole=judge.role) (cost=0.25 rows=1) (actual time=0.007..0.007 rows=0 loops=2808)

## Query#1b

Performance with an index on the judge's role in a competition/program. We chose to try indexing on this attribute because it is used in the operation to JOIN the judge and score that was assigned to a player by a particular judge. Since JOIN is an expensive operation, we wanted to see if we could reduce runtime for finding attributes used in the join. We can see that this was a good idea, as it decreased the runtime immensely from 32ms to just 9ms

**Schema SQL**

```
LECLAVELIER, FKA );
877 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie
GOLD', 'USA');
878 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka
NAGAI', 'JPN');
879 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao
ASADA', 'JPN');
880 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia
GALUSTYAN', 'ARM');
881 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena
LEONOVA', 'RUS');
882
883 CREATE INDEX idx_judge_role ON judge(role);
884 # CREATE INDEX idx_judge_comp ON judge(competition);
```

Text to DDL

**Query SQL**

```
1 -- average score judge gives skaters of different nationalities
2
3 EXPLAIN ANALYZE SELECT judgeName, judge.nationality AS judgeNat,
skater.nationality AS skaterNat, AVG(avgComponentScore) AS
avgScoreFromJudge
4 FROM skater JOIN score ON name = skaterName JOIN judge ON
competitionName = competition AND score.program = judge.program
AND judgeRole = role
5 GROUP BY judgeName, judge.nationality, skater.nationality;
6
7 -- select the highest and lowest avg score a judge gives to a
nationality and compare to average score they give per
nationality
8
```

Results

Copy as Markdown

Query #1

Execution time: 9ms

EXPLAIN

-> Table scan on <temporary> (actual time=8.071..8.103 rows=226 loops=1) -> Aggregate using temporary table (actual time=8.069..8.069 rows=226 loops=1) -> Nested loop inner join (cost=41613.25 rows=5616) (actual time=0.128..7.110 rows=702 loops=1) -> Nested loop inner join (cost=41032.15 rows=5616) (actual time=0.122..6.696 rows=702 loops=1) -> Table scan on score (cost=70.45 rows=702) (actual time=0.093..0.385 rows=702 loops=1) -> Index lookup on judge using idx\_judge\_role (role=score.judgeRole), with index condition: ((judge.competition = score.competitionName) and (score.program = judge.program)) (cost=0.75 rows=8) (actual time=0.008..0.009 rows=1 loops=702) -> Single-row index lookup on skater using PRIMARY (name=score.skaterName) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=702)

## Query#1c

Because of the promising results of the indexing on skater nationality, we wanted to see if indexing with the rest of the group-by attributes would have any benefits. Unfortunately, no combination of group-by attributes (judge name, judge nationality and skater nationality) resulted in better performance than 24ms. Additional combinations we tried were skater nationality and judge role (since judge role led to such improvement), but including other indexes with judge role led to slower performance than just judge role by itself, as shown below with 18ms.



## Schema SQL ●

```
070 INSERT IGNORE INTO skater(name,nationality) VALUES ('Toku
NAGAI', 'JPN');
879 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao
ASADA', 'JPN');
880 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia
GALUSTYAN', 'ARM');
881 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena
LEONOVA', 'RUS');
882
883 # CREATE INDEX idx_judge_name ON judge(judgeName);
884 # CREATE INDEX idx_judge_nat ON judge(nationality);
885 CREATE INDEX idx_skater_nat ON skater(nationality);
886 CREATE INDEX idx_j_role ON judge(role);
887
```

Text to DDL

## Query SQL ●

```
1 -- average score judge gives skaters of different nationalities
2
3 EXPLAIN ANALYZE SELECT judgeName, judge.nationality AS judgeNat,
skater.nationality AS skaterNat, AVG(avgComponentScore) AS
avgScoreFromJudge
4 FROM skater JOIN score ON name = skaterName JOIN judge ON
competitionName = competition AND score.program = judge.program
AND judgeRole = role
5 GROUP BY judgeName, judge.nationality, skater.nationality;
6
7 -- select the highest and lowest avg score a judge gives to a
nationality and compare to average score they give per
nationality
8
```

## Results

Copy as Markdown

Query #1 Execution time: 18ms

## EXPLAIN

-> Table scan on <temporary> (actual time=8.299..8.333 rows=226 loops=1) -> Aggregate using temporary table (actual time=8.297..8.297 rows=226 loops=1) -> Nested loop inner join (cost=41613.25 rows=5616) (actual time=0.126..7.318 rows=702 loops=1) -> Nested loop inner join (cost=41032.15 rows=5616) (actual time=0.120..6.896 rows=702 loops=1) -> Table scan on score (cost=70.45 rows=702) (actual time=0.092..0.395 rows=702 loops=1) -> Index lookup on judge using idx\_j\_role (role=score.judgeRole), with index condition: ((judge.competition = score.competitionName) and (score.program = judge.program)) (cost=0.75 rows=8) (actual time=0.009..0.009 rows=1 loops=702) -> Single-row index lookup on skater using PRIMARY (name=score.skaterName) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=702)

## Schema SQL ●

```
871 INSERT IGNORE INTO skater(name,nationality) VALUES ('Maria
SOTSKOVA', 'RUS');
872 INSERT IGNORE INTO skater(name,nationality) VALUES ('Wakaba
HIGUCHI', 'JPN');
873 INSERT IGNORE INTO skater(name,nationality) VALUES ('So Youn
PARK', 'KOR');
874 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mae Berenice
MEITE', 'FRA');
875 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabrielle
DALEMAN', 'CAN');
876 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine
LECAVELIER', 'FRA');
877 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie
GOLD', 'USA');
```

Text to DDL

## Query SQL ●

```
1 -- average score judge gives skaters of different nationalities
2
3 EXPLAIN ANALYZE SELECT judgeName, judge.nationality AS judgeNat,
skater.nationality AS skaterNat, AVG(avgComponentScore) AS
avgScoreFromJudge
4 FROM skater JOIN score ON name = skaterName JOIN judge ON
competitionName = competition AND score.program = judge.program
AND judgeRole = role
5 GROUP BY judgeName, judge.nationality, skater.nationality;
6
7 -- select the highest and lowest avg score a judge gives to a
nationality and compare to average score they give per
nationality
8
```

## Results

Copy as Markdown

Query #1 Execution time: 29ms

## EXPLAIN

-> Table scan on <temporary> (actual time=27.538..27.573 rows=226 loops=1) -> Aggregate using temporary table (actual time=27.537..27.537 rows=226 loops=1) -> Nested loop inner join (cost=1268.01 rows=2808) (actual time=0.418..26.443 rows=702 loops=1) -> Inner hash join (no condition) (cost=285.21 rows=2808) (actual time=0.328..0.689 rows=2808 loops=1) -> Table scan on judge (cost=0.19 rows=72) (actual time=0.070..0.115 rows=72 loops=1) -> Hash -> Covering index scan on skater using idx\_skater\_nat (cost=4.15 rows=39) (actual time=0.169..0.175 rows=39 loops=1) -> Filter: (score.program = judge.program) (cost=0.25 rows=1) (actual time=0.009..0.009 rows=0 loops=2808) -> Single-row index lookup on score using PRIMARY (skaterName=skater.name, competitionName=judge.competition, program=judge.program, judgeRole=judge.role) (cost=0.25 rows=1) (actual time=0.009..0.009 rows=0 loops=2808)

## Query #2

Performance without indices. (Runtime: 28.47 ms.)

Schema SQL

```
100 -- select the highest and lowest avg score a judge gives to a nationality and compare to average
101 -- score they give per nationality
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Text to DDL

Query SQL

```
1 EXPLAIN ANALYZE SELECT judgeName, MAX(avgScoreFromJudge), MIN(avgScoreFromJudge),
2   AVG(avgScoreFromJudge)
3 FROM (
4   SELECT judgeName, judge.nationality AS judgeNat, skater.nationality AS skaterNat,
5     AVG(avgComponentScore) AS avgScoreFromJudge
6   FROM skater JOIN score ON name = skaterName JOIN judge ON competitionName = competition AND
7     score.program = judge.program AND judgeRole = role
8   GROUP BY judgeName, judge.nationality, skater.nationality) avgs
9 GROUP BY judgeName
```

Copy as Markdown

Results

Query #1 Execution time: 32ms

EXPLAIN

-> Table scan on <temporary> (actual time=28.469..28.472 rows=23 loops=1) -> Aggregate using temporary table (actual time=28.468..28.468 rows=23 loops=1) -> Table scan on avgs (cost=2.50..2.50 rows=0) (actual time=28.178..28.211 rows=226 loops=1) -> Materialize (cost=2.50..2.50 rows=0) (actual time=28.177..28.177 rows=226 loops=1) -> Table scan on <temporary> (actual time=27.932..27.980 rows=226 loops=1) -> Aggregate using temporary table (actual time=27.930..27.930 rows=226 loops=1) -> Nested loop inner join (cost=1268.01 rows=2808) (actual time=0.320..0.320 rows=702 loops=1) -> Inner hash join (no condition) (cost=285.21 rows=2808) (actual time=0.246..0.602 rows=2808 loops=1) -> Table scan on judge (cost=0.19 rows=72) (actual time=0.177..0.238 rows=72 loops=1) -> Hash -> Table scan on skater (cost=4.15 rows=39) (actual time=0.039..0.047 rows=39 loops=1) -> Filter: (score.program = judge.program) (cost=0.25 rows=1) (actual time=0.009..0.009 rows=0 loops=2808) -> Single-row index lookup on score using PRIMARY (skaterName=skater.name, competitionName=judge.competition, program=judge.program, judgeRole=judge.role) (cost=0.25 rows=1) (actual time=0.009..0.009 rows=0 loops=2808)

## Query #2a

Performance with index on skater nationality. (Runtime: 24.44 ms.) We create an index on the skater(nationality) attribute, as this is being called in the inner SELECT function. As seen when comparing the two queries of EXPLAIN ANALYZE on the default index as well as the newly created index for the skater(nationality), the latter query has a faster runtime than the former... aka, an index for the skater nationality really benefits the runtime for this query.

Schema SQL

```
100 -- select the highest and lowest avg score a judge gives to a nationality and compare to average
101 -- score they give per nationality
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Text to DDL

Query SQL

```
1 EXPLAIN ANALYZE SELECT judgeName, MAX(avgScoreFromJudge), MIN(avgScoreFromJudge),
2   AVG(avgScoreFromJudge)
3 FROM (
4   SELECT judgeName, judge.nationality AS judgeNat, skater.nationality AS skaterNat,
5     AVG(avgComponentScore) AS avgScoreFromJudge
6   FROM skater JOIN score ON name = skaterName JOIN judge ON competitionName = competition AND
7     score.program = judge.program AND judgeRole = role
8   GROUP BY judgeName, judge.nationality, skater.nationality) avgs
9 GROUP BY judgeName
```

Copy as Markdown

Results

Query #1 Execution time: 29ms

EXPLAIN

-> Table scan on <temporary> (actual time=24.442..24.445 rows=23 loops=1) -> Aggregate using temporary table (actual time=24.441..24.441 rows=23 loops=1) -> Table scan on avgs (cost=2.50..2.50 rows=0) (actual time=24.123..24.157 rows=226 loops=1) -> Materialize (cost=2.50..2.50 rows=0) (actual time=24.122..24.122 rows=226 loops=1) -> Table scan on <temporary> (actual time=23.801..23.852 rows=226 loops=1) -> Aggregate using temporary table (actual time=23.799..23.799 rows=226 loops=1) -> Nested loop inner join (cost=1268.01 rows=2808) (actual time=0.348..0.348 rows=702 loops=1) -> Inner hash join (no condition) (cost=285.21 rows=2808) (actual time=0.233..0.635 rows=2808 loops=1) -> Table scan on judge (cost=0.19 rows=72) (actual time=0.161..0.215 rows=72 loops=1) -> Hash -> Covering index scan on skater using index\_skater\_nat (cost=4.15 rows=39) (actual time=0.042..0.049 rows=39 loops=1) -> Filter: (score.program = judge.program) (cost=0.25 rows=1) (actual time=0.008..0.008 rows=0 loops=2808) -> Single-row index lookup on score using PRIMARY (skaterName=skater.name, competitionName=judge.competition, program=judge.program, judgeRole=judge.role) (cost=0.25 rows=1) (actual time=0.005..0.005 rows=0 loops=2808)

## Query #2b

Performance with index on judge program. (Runtime: 17.52 ms.) We create an index on the judge(program) attribute, as this is being called in the FROM function; the multiple tables are being joined based on the judge.program being equal to score.program, ensuring that they are both from the same discipline/event type (e.g., Men's Short Program). We can see that similar to Query #2a, an index for the judge program really benefits the runtime for this query (even more so than Query #2a).

Schema SQL

```
871 INSERT IGNORE INTO skater(name,nationality) VALUES ('Maria SOTSKOVA','RUS');
872 INSERT IGNORE INTO skater(name,nationality) VALUES ('Makaba HIGUCHI','JPN');
873 INSERT IGNORE INTO skater(name,nationality) VALUES ('So Youn PARK','KOR');
874 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mae Berenice HEITZ','FRA');
875 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabrielle DALEMAN','CAN');
876 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine LECACHELIER','FRA');
877 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gracie GOLD','USA');
878 INSERT IGNORE INTO skater(name,nationality) VALUES ('Yuka NAGAI','JPN');
879 INSERT IGNORE INTO skater(name,nationality) VALUES ('Mao ASADA','JPN');
880 INSERT IGNORE INTO skater(name,nationality) VALUES ('Anastasia GALUSTYAN','ARM');
881 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena LEONOVA','RUS');
882
883 -- select the highest and lowest avg score a judge gives to a nationality and compare to average
884 score they give per nationality
885 CREATE INDEX index_judge_program ON judge(program);
```

Text to DDL

Query SQL

```
1 EXPLAIN ANALYZE SELECT judgeName, MAX(avgScoreFromJudge), MIN(avgScoreFromJudge),
  AVG(avgScoreFromJudge)
2 FROM (
3   SELECT judgeName, judge.nationality AS judgeNat, skater.nationality AS skaterNat,
  AVG(avgComponentScore) AS avgScoreFromJudge
4   FROM skater JOIN score ON name = skaterName JOIN judge ON competitionName = competition AND
  score.program = judge.program AND judgeRole = role
5   GROUP BY judgeName, judge.nationality, skater.nationality) avgs
6 GROUP BY judgeName
```

Copy as Markdown

Results

Query #1

Execution time: 19ms

Copy as Markdown

EXPLAIN

```
--> Table scan on <temporary> (actual time=17.521..17.524 rows=23 loops=1) --> Aggregate using temporary table (actual time=17.520..17.520 rows=23 loops=1) --> Table scan on avgs (cost=2.50..2.50 rows=0) (actual time=17.267..17.301 rows=226 loops=1) --> Materialize
(cost=2.50..2.50 rows=0) (actual time=17.265..17.265 rows=226 loops=1) --> Table scan on <temporary> (actual time=8.383..8.438 rows=226 loops=1) --> Aggregate using temporary table (actual time=8.381..8.381 rows=226 loops=1) --> Nested loop inner join (cost=46740.29
(cost=6318) (actual time=0.157..7.349 rows=702 loops=1) --> Nested loop inner join (cost=46086.55 rows=6318) (actual time=0.150..6.927 rows=702 loops=1) --> Table scan on score (cost=70.45 rows=702) (actual time=0.117..0.433 rows=702 loops=1) --> Index lookup on judge using
index_judge_program (program=score.program), with index condition: (judge.role = score.judgeRole) and (judge.competition = score.competitionName) and (score.program = judge.program) (cost=0.75 rows=9) (actual time=0.008..0.009 rows=1 loops=702) --> Single-row index
lookup on skater using PRIMARY (name=score.skaterName) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=702)
```

## Query #2c

Performance with index on the average component score. (Runtime: 32.19 ms.) We create an index on the score(avgComponentScore), as this is being called in the inner SELECT function. Unfortunately, it seems that an index on this particular attribute slows the runtime by approximately ~4 ms, which is not ideal. It is possible that this occurs because every score will have a unique component score, ranging from 0 - 10 with a 0.25 step. (E.g., 8.00, 8.25, 8.50, 8/75, etc.)

Schema SQL

```
849 INSERT IGNORE INTO skater(name,nationality) VALUES ('Denis TEN','KAZ');
850 INSERT IGNORE INTO skater(name,nationality) VALUES ('Nathan CHEN','USA');
851 INSERT IGNORE INTO skater(name,nationality) VALUES ('Takahito MURA','JPN');
852 INSERT IGNORE INTO skater(name,nationality) VALUES ('Misha GE','UZB');
853 INSERT IGNORE INTO skater(name,nationality) VALUES ('Artur DMITRIYEV','RUS');
854 INSERT IGNORE INTO skater(name,nationality) VALUES ('Jorik HENDRICKX','BEL');
855 INSERT IGNORE INTO skater(name,nationality) VALUES ('Chafik BESSEGHIER','FRA');
856 INSERT IGNORE INTO skater(name,nationality) VALUES ('Brendan KERRY','AUS');
857 INSERT IGNORE INTO skater(name,nationality) VALUES ('Ivan RIGHINI','ITA');
858 INSERT IGNORE INTO skater(name,nationality) VALUES ('Evgenia MEDVEDEVA','RUS');
859 INSERT IGNORE INTO skater(name,nationality) VALUES ('Gabrielle DALEMAN','CAN');
860 INSERT IGNORE INTO skater(name,nationality) VALUES ('Maria SOTSKOVA','RUS');
861 INSERT IGNORE INTO skater(name,nationality) VALUES ('Laurine LECACHELIER','FRA');
862 INSERT IGNORE INTO skater(name,nationality) VALUES ('Makaba HIGUCHI','JPN');
863 INSERT IGNORE INTO skater(name,nationality) VALUES ('So Youn PARK','KOR');
864 INSERT IGNORE INTO skater(name,nationality) VALUES ('Alena LEONOVA','RUS');
```

Text to DDL

Query SQL

```
1 EXPLAIN ANALYZE SELECT judgeName, MAX(avgScoreFromJudge), MIN(avgScoreFromJudge),
  AVG(avgScoreFromJudge)
2 FROM (
3   SELECT judgeName, judge.nationality AS judgeNat, skater.nationality AS skaterNat,
  AVG(avgComponentScore) AS avgScoreFromJudge
4   FROM skater JOIN score ON name = skaterName JOIN judge ON competitionName = competition AND
  score.program = judge.program AND judgeRole = role
5   GROUP BY judgeName, judge.nationality, skater.nationality) avgs
6 GROUP BY judgeName
```

Copy as Markdown

Results

Query #1

Execution time: 33ms

Copy as Markdown

EXPLAIN

```
--> Table scan on <temporary> (actual time=32.192..32.211 rows=23 loops=1) --> Aggregate using temporary table (actual time=32.192..32.192 rows=23 loops=1) --> Table scan on avgs (cost=2.50..2.50 rows=0) (actual time=31.835..31.870 rows=226 loops=1) --> Materialize
(cost=2.50..2.50 rows=0) (actual time=31.834..31.834 rows=226 loops=1) --> Table scan on <temporary> (actual time=31.559..31.611 rows=226 loops=1) --> Aggregate using temporary table (actual time=31.556..31.556 rows=226 loops=1) --> Nested loop inner join (cost=1268.01
rows=2808) (actual time=0.121..30.315 rows=702 loops=1) --> Inner hash join (no condition) (cost=285.21 rows=2808) (actual time=0.070..0.399 rows=2808 loops=1) --> Table scan on judge (cost=0.19 rows=72) (actual time=0.022..0.076 rows=72 loops=1) --> Hash --> Table scan on
skater (cost=4.15 rows=39) (actual time=0.029..0.035 rows=39 loops=1) --> Filter: (score.program = judge.program) (cost=0.25 rows=1) (actual time=0.010..0.011 rows=0 loops=2808) --> Single-row index lookup on score using PRIMARY (skaterName=skater.name,
competitionName=judge.competition, program=judge.program, judgeRole=judge.role) (cost=0.25 rows=1) (actual time=0.010..0.010 rows=0 loops=2808)
```