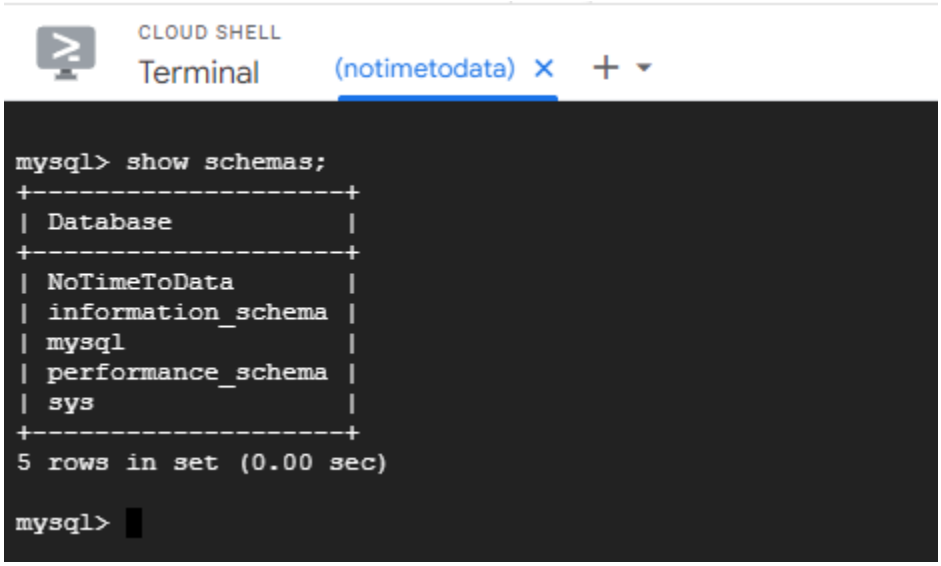
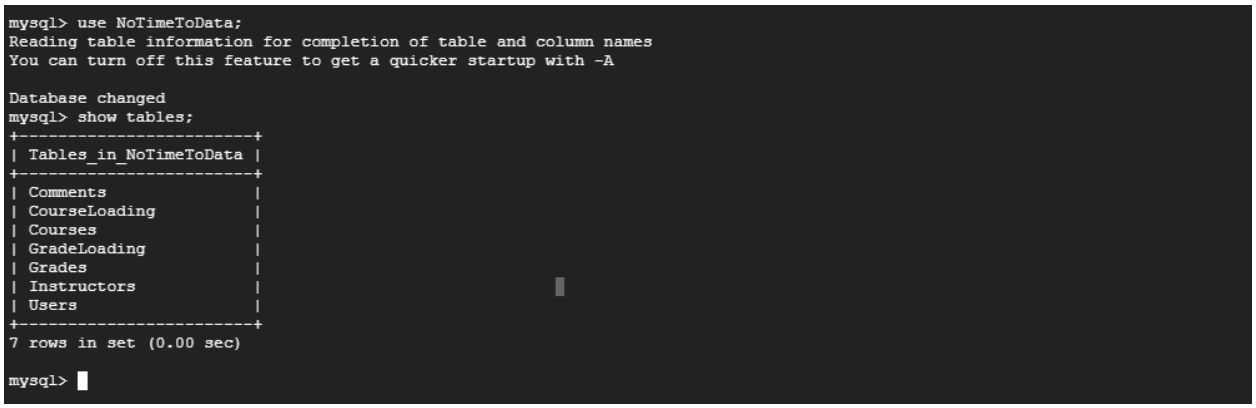
GCP Implementation:

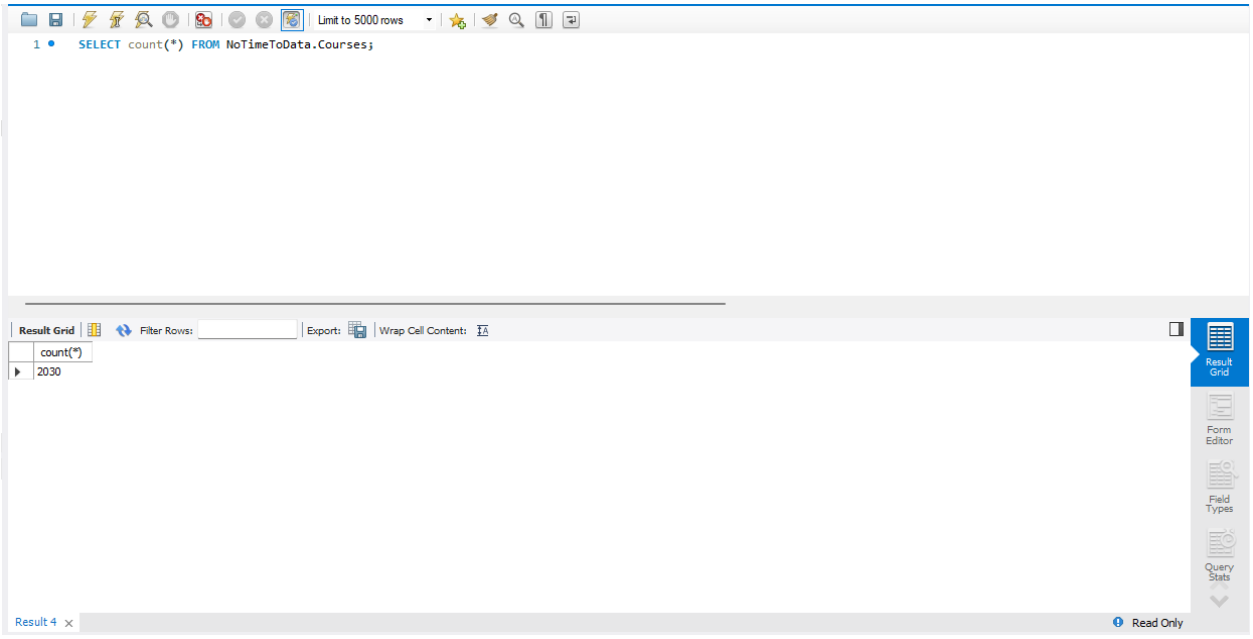


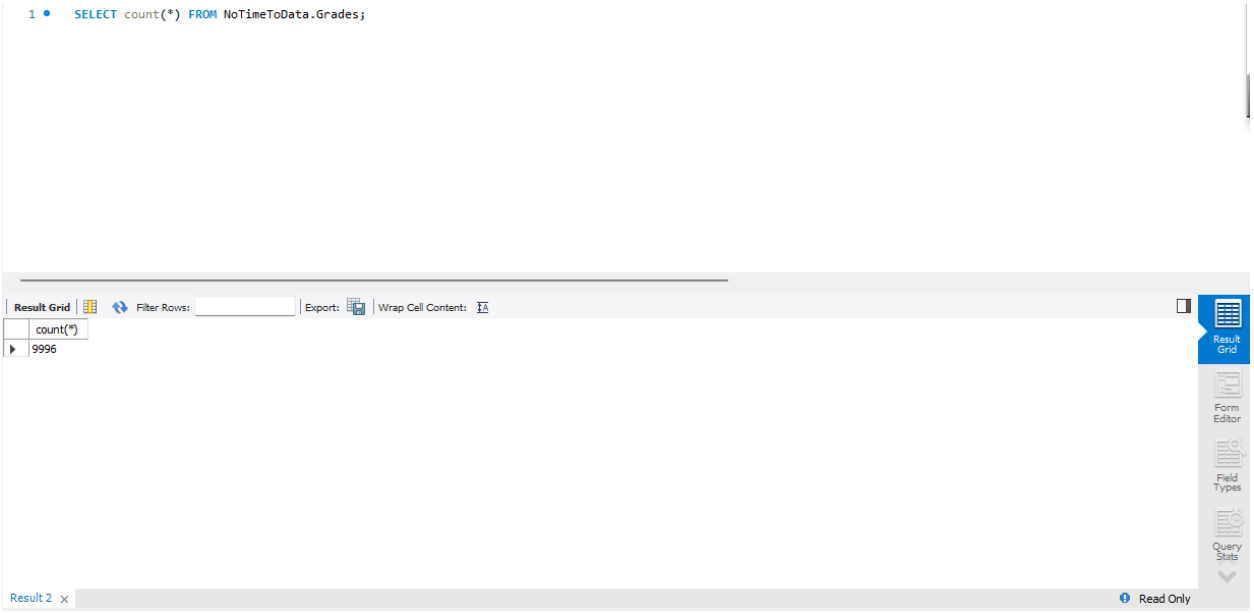


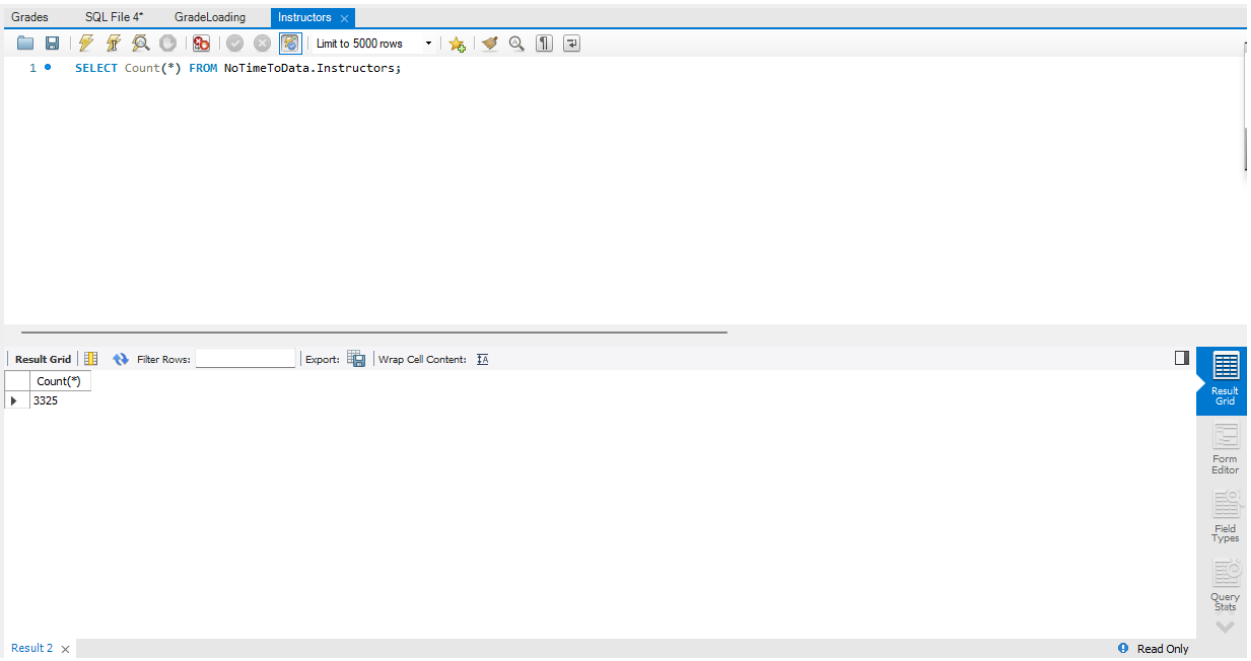
DDL Commands:

```sql
DROP TABLE IF EXISTS Instructors, Courses, Grades, Comments, Users;

CREATE TABLE Instructors(
    firstName  VARCHAR(50),
    lastName   VARCHAR(50),
    primary key(firstName, lastName)
);

CREATE TABLE Courses(
    year         INT,
    term         VARCHAR(10),
    yearTerm     VARCHAR(10),
    subject       VARCHAR(10),
    cNumber      INT,
    acp          VARCHAR(10),
    cs           VARCHAR(10),
    hum          VARCHAR(10),
    nat          VARCHAR(10),
    qr           VARCHAR(10),
    sbs          VARCHAR(10),
    primary key(yearTerm, subject, cNumber)
);

CREATE TABLE Grades(
    yearTerm     VARCHAR(10) references Courses(yearTerm),
    subject      VARCHAR(10) references Courses(subject),
    cNumber      INT references Courses(cNumber),
    courseName   VARCHAR(255),
    schedType    VARCHAR(10),
    insLastName  VARCHAR(50) references Instructors(lastName),
    insFirstName VARCHAR(50) references Instructors(firstName),
    aPlus        INT,
    a            INT,
    aMinus       INT,
    bPlus        INT,
    b            INT,
    bMinus       INT,
    cPlus        INT,
    c            INT,
    cMinus       INT,
    dPlus        INT,
    d            INT,
    dMinus       INT,
    f            INT,
    w            INT,
    primary key(yearTerm, subject, cNumber, courseName, schedType, insLastName, insFirstName)
);

CREATE TABLE Comments(
    commentID    VARCHAR(20) Primary key,
    content      VARCHAR(255),
    email        VARCHAR(255) references Users(email),
    yearTerm     VARCHAR(10) references Courses(yearTerm),
    subject      VARCHAR(10) references Courses(subject),
    cNumber      INT references Courses(cNumber)
);

CREATE TABLE Users(
    email      VARCHAR(255) primary key,
    password   VARCHAR(18)
);

-- CREATE TABLE Teach(
--     yearTerm     VARCHAR(10) references Courses(yearTerm),
--     subject      VARCHAR(10) references Courses(subject),
--     cNumber      INT references Courses(cNumber),
--  netId        VARCHAR(20) references Instructors(netId),
--  primary key(yearTerm, subject, cNumber)
-- );
```

Inserting 1000+ rows: Count Query:



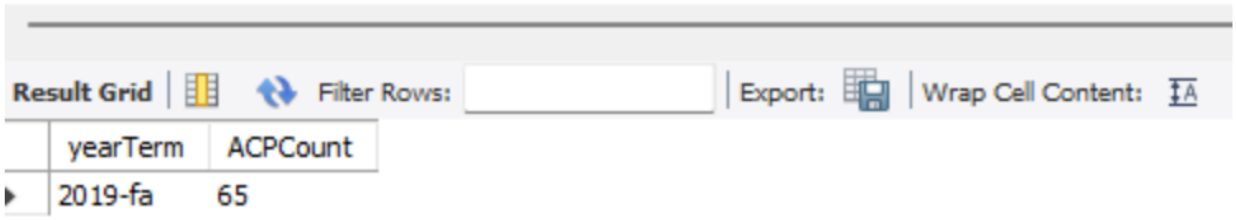



Grades
SQL File 4*
GradeLoading
Instructors
Limit to 5000 rows
1
SELECT Count(*) FROM NoTimeToData.Instructors;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Count(*)
3325
Result Grid
Form Editor
Field Types
Query Stats
Result 2
Read Only

Advanced Query:

Advanced Query 1:

```
1    -- ACP Count
2    SELECT yearTerm, Count(*) as ACPCount
3    FROM Courses c NATURAL JOIN Grades g
4    WHERE acp != ''
5    GROUP BY yearTerm;
```



yearTerm	ACPCount
2019-fa	65

Advanced Query 2:

```
1 •  SELECT subject, cNumber, courseName,
2    (sum(aPlus)+sum(a))/
3    (sum(aPlus)+sum(a)+sum(aMinus)+sum(bPlus)+sum(b)+sum(bMinus)+sum(cPlus)+sum(c)+sum(cMinus)+sum(dPLus)+sum(d)+sum(dMinus)+sum(f)+sum(w))
4    FROM NoTimeToData.Grades
5    Group by subject, cNumber, courseName
6    LIMIT 20
```

100% 9:6

Result Grid Filter Rows: Search Export: Fetch rows:

subject	cNumber	courseName	(sum(aPlus)+sum(a))/
AAS	100	Intro Asian American Studies	0.5687
AAS	246	Asian American Youth in Film	0.8269
AAS	287	Food and Asian Americans	0.8235
AAS	370	Immigration, Law, and Rights	0.1034
ABE	100	Intro Agric & Biological Engrg	0.8000
ABE	199	Water in the Global Environ	0.9592
ABE	223	ABE Principles: Machine Syst	0.5349
ABE	224	ABE Principles: Soil & Water	0.6667
ABE	430	Project Management	0.7698
ABE	436	Renewable Energy Systems	0.3248
ABE	488	Bioprocessing Biomass for…	0.3544
ABE	498	Eng Design for Solar Smart…	0.7561
ACCY	200	Fundamentals of Accounting	0.3359
ACCY	201	Accounting and Accountancy I	0.2674
ACCY	202	Accounting and Accountanc…	0.3219
ACCY	301	Atg Measurement & Disclos…	0.2767
ACCY	302	Decision Making for Atg	0.3209
ACCY	303	Atg Institutions and Reg	0.4362
ACCY	304	Accounting Control Systems	0.3015
ACCY	312	Principles of Taxation	0.3155

Indexing:

Advanced Query 1:

```sql
-- ACP Count
SELECT yearTerm, Count(*) as ACPCount
FROM Courses c NATURAL JOIN Grades g
WHERE acp != ''
GROUP BY yearTerm;
```

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

yearTerm	ACPCount
2019-fa	65

Advanced Query 1: Before Indexing:

```
mysql> EXPLAIN ANALYZE SELECT yearTerm, Count(*) as ACPCount FROM Courses c NATURAL JOIN Grades g WHERE acp != '' GROUP BY yearTerm;
+------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                  |
+------------------------------------------------------------------------------------------+
| -> Group aggregate: count(0)  (cost=1196.54 rows=2423) (actual time=1.708..1.708 rows=1 loops=1)
    -> Nested loop inner join  (cost=954.23 rows=2423) (actual time=0.854..1.694 rows=65 loops=1)
        -> Filter: (c.acp <> '')  (cost=205.50 rows=1827) (actual time=0.054..0.788 rows=288 loops=1)
            -> Index scan on c using PRIMARY  (cost=205.50 rows=2030) (actual time=0.045..0.640 rows=2030 loops=1)
        -> Index lookup on g using PRIMARY (yearTerm=c.yearTerm, subject=c.`subject`, cNumber=c.cNumber)  (cost=0.28 rows=1) (actual time=0.003..0.003 rows=0 loops=288)
 |
+------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Advanced Query 1: After indexing

```
mysql> CREATE INDEX courses_cNumber_idx on Courses(cNumber);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT yearTerm, Count(*) as ACPCount FROM Courses c NATURAL JOIN Grades g WHERE acp != '' GROUP BY yearTerm;
+--------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                    |
+--------------------------------------------------------------------------------------------------------------------------------------------+
| -> Group aggregate: count(0)  (cost=1196.54 rows=2423) (actual time=1.597..1.597 rows=1 loops=1)
    -> Nested loop inner join  (cost=954.23 rows=2423) (actual time=0.774..1.584 rows=65 loops=1)
        -> Filter: (c.acp <> '')  (cost=205.50 rows=1827) (actual time=0.052..0.737 rows=288 loops=1)
            -> Index scan on c using PRIMARY  (cost=205.50 rows=2030) (actual time=0.043..0.590 rows=2030 loops=1)
        -> Index lookup on g using PRIMARY (yearTerm=c.yearTerm, subject=c.`subject`, cNumber=c.cNumber)  (cost=0.28 rows=1) (actual time=0.003..0.003 rows=0 loops=288)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> CREATE INDEX courses_subject_idx on Courses(subject);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT yearTerm, Count(*) as ACPCount FROM Courses c NATURAL JOIN Grades g WHERE acp != '' GROUP BY yearTerm;
+--------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                    |
+--------------------------------------------------------------------------------------------------------------------------------------------+
| -> Group aggregate: count(0)  (cost=1196.54 rows=2423) (actual time=1.907..1.908 rows=1 loops=1)
    -> Nested loop inner join  (cost=954.23 rows=2423) (actual time=0.973..1.891 rows=65 loops=1)
        -> Filter: (c.acp <> '')  (cost=205.50 rows=1827) (actual time=0.162..0.929 rows=288 loops=1)
            -> Index scan on c using PRIMARY  (cost=205.50 rows=2030) (actual time=0.152..0.771 rows=2030 loops=1)
        -> Index lookup on g using PRIMARY (yearTerm=c.yearTerm, subject=c.`subject`, cNumber=c.cNumber)  (cost=0.28 rows=1) (actual time=0.003..0.003 rows=0 loops=288)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> CREATE INDEX courses_acp_idx on Courses(acp);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT yearTerm, Count(*) as ACPCount FROM Courses c NATURAL JOIN Grades g WHERE acp != '' GROUP BY yearTerm;
+--------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                    |
+--------------------------------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.001..0.001 rows=1 loops=1)
    -> Aggregate using temporary table  (actual time=1.049..1.049 rows=1 loops=1)
        -> Nested loop inner join  (cost=180.77 rows=383) (actual time=0.430..0.989 rows=65 loops=1)
            -> Filter: (c.acp <> '')  (cost=62.34 rows=289) (actual time=0.077..0.176 rows=288 loops=1)
                -> Index range scan on c using courses_acp_idx  (cost=62.34 rows=289) (actual time=0.037..0.107 rows=288 loops=1)
            -> Index lookup on g using PRIMARY (yearTerm=c.yearTerm, subject=c.`subject`, cNumber=c.cNumber)  (cost=0.28 rows=1) (actual time=0.003..0.003 rows=0 loops=288)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

We chose subject, cNumber, and acp as our 3 different indexing strategies. Since we would like to see how the data are distributed across subjects and course numbers. The performance shows that the database can search more efficiently based on cNumber itself. Since the given query is built based on ACP, the acp index outperforms the subject index and cNumber index.

Advanced Query 2:

```sql
SELECT subject, cNumber, courseName,
(sum(aPlus)+sum(a))/
(sum(aPlus)+sum(a)+sum(aMinus)+sum(bPlus)+sum(b)+sum(bMinus)+sum(cPlus)+sum(c)+sum(cMinus)+sum(dPLus)+sum(d)+sum(dMinus)+sum(f)+sum(w))
FROM NoTimeToData.Grades
Group by subject, cNumber, courseName
LIMIT 20
```

100% 9:6

Result Grid Filter Rows: Search Export: Fetch rows:

subject	cNumber	courseName	(sum(aPlus)+sum(a))/
AAS	100	Intro Asian American Studies	0.5687
AAS	246	Asian American Youth in Film	0.8269
AAS	287	Food and Asian Americans	0.8235
AAS	370	Immigration, Law, and Rights	0.1034
ABE	100	Intro Agric & Biological Engrg	0.8000
ABE	199	Water in the Global Environ	0.9592
ABE	223	ABE Principles: Machine Syst	0.5349
ABE	224	ABE Principles: Soil & Water	0.6667
ABE	430	Project Management	0.7698
ABE	436	Renewable Energy Systems	0.3248
ABE	488	Bioprocessing Biomass for...	0.3544
ABE	498	Eng Design for Solar Smart...	0.7561
ACCY	200	Fundamentals of Accounting	0.3359
ACCY	201	Accounting and Accountancy I	0.2674
ACCY	202	Accounting and Accountanc...	0.3219
ACCY	301	Atg Measurement & Disclos...	0.2767
ACCY	302	Decision Making for Atg	0.3209
ACCY	303	Atg Institutions and Reg	0.4362
ACCY	304	Accounting Control Systems	0.3015
ACCY	312	Principles of Taxation	0.3155

```
mysql> EXPLAIN ANALYZE SELECT subject, cNumber, courseName, (sum(aPlus)+sum(a))/(sum(aPlus)+sum(a)+sum(aMinus)+sum(bPlus)+sum(b)+sum(bMinus)+sum(cPlus)+sum(c)+sum(cMinus)+sum(dPLus)+sum(d)+sum(dMinus)+sum(f)+sum(w))  FROM NoTimeToData.Grades
Group by subject, cNumber, courseName LIMIT 20;
+--------------------------------------------------------------+
| EXPLAIN                                                      |
+--------------------------------------------------------------+
| -> Limit: 20 row(s)  (actual time=40.286..40.298 rows=20 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.011 rows=20 loops=1)
        -> Aggregate using temporary table  (actual time=40.285..40.295 rows=20 loops=1)
            -> Table scan on Grades  (cost=1039.85 rows=9996) (actual time=0.130..8.195 rows=9996 loops=1)
 |
+--------------------------------------------------------------+
1 row in set (0.05 sec)
```

```
mysql> CREATE INDEX grades_subject_idx on Grades(subject);
Query OK, 0 rows affected (0.29 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT subject, cNumber, courseName, (sum(aPlus)+sum(a))/(sum(aPlus)+sum(a)+sum(aMinus)+sum(bPlus)+sum(b)+sum(bMinus)+sum(cPlus)+sum(c)+sum(cMinus)+sum(dPLus)+sum(d)+sum(dMinus)+sum(f)+sum(w))  FROM NoTimeToData.Grades
Group by subject, cNumber, courseName;
+--------------------------------------------------------------+
| EXPLAIN                                                      |
+--------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.002..1.283 rows=3105 loops=1)
    -> Aggregate using temporary table  (actual time=36.071..37.540 rows=3105 loops=1)
        -> Table scan on Grades  (cost=1039.85 rows=9996) (actual time=0.100..7.676 rows=9996 loops=1)
 |
+--------------------------------------------------------------+
1 row in set (0.05 sec)
```

```
mysql> CREATE INDEX grades_cnumber_idx on Grades(cnumber);
Query OK, 0 rows affected (0.26 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT subject, cNumber, courseName, (sum(aPlus)+sum(a))/(sum(aPlus)+sum(a)+sum(aMinus)+sum(bPlus)+sum(b)+sum(bMinus)+sum(cPlus)+sum(c)+sum(cMinus)+sum(dPLus)+sum(d)+sum(dMinus)+sum(f)+sum(w))  FROM NoTimeToData.Grades
Group by subject, cNumber, courseName;
+--------------------------------------------------------------+
| EXPLAIN                                                      |
+--------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.002..1.364 rows=3105 loops=1)
    -> Aggregate using temporary table  (actual time=36.940..38.488 rows=3105 loops=1)
        -> Table scan on Grades  (cost=1039.85 rows=9996) (actual time=0.096..7.753 rows=9996 loops=1)
 |
+--------------------------------------------------------------+
1 row in set (0.04 sec)
```

```
mysql> CREATE INDEX grades_ins_idx on Grades(insLastName, insFirstName);
Query OK, 0 rows affected (0.29 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT subject, cNumber, courseName, (sum(aPlus)+sum(a))/(sum(aPlus)+sum(a)+sum(aMinus)+sum(bPlus)+sum(b)+sum(bMinus)+sum(cPlus)+sum(c)+sum(cMinus)+sum(dPLus)+sum(d)+sum(dMinus)+sum(f)+sum(w))  FROM NoTimeToData.Grades
Group by subject, cNumber, courseName;
+--------------------------------------------------------------+
| EXPLAIN                                                      |
+--------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.001..1.298 rows=3105 loops=1)
    -> Aggregate using temporary table  (actual time=36.297..37.885 rows=3105 loops=1)
        -> Table scan on Grades  (cost=1039.85 rows=9996) (actual time=0.176..7.862 rows=9996 loops=1)
 |
+--------------------------------------------------------------+
1 row in set (0.04 sec)
```

```
mysql> CREATE INDEX grades_mix_idx on Grades(subject,cnumber,yearTerm);
ERROR 1061 (42000): Duplicate key name 'grades_mix_idx'
mysql> EXPLAIN ANALYZE SELECT subject, cNumber, courseName, (sum(aPlus)+sum(a))/(sum(aPlus)+sum(a)+sum(aMinus)+sum(bPlus)+sum(b)+sum(bMinus)+sum(cPlus)+sum(c)+sum(cMinus)+sum(dPLus)+sum(d)+sum(dMinus)+sum(f)+sum(w))  FROM NoTimeToData.Grades
Group by subject, cNumber, courseName;
+--------------------------------------------------------------+
| EXPLAIN                                                      |
+--------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.002..1.270 rows=3105 loops=1)
    -> Aggregate using temporary table  (actual time=37.550..38.999 rows=3105 loops=1)
        -> Table scan on Grades  (cost=1039.85 rows=9996) (actual time=0.099..7.909 rows=9996 loops=1)
 |
+--------------------------------------------------------------+
1 row in set (0.04 sec)
```

For the second query, we wanted to see how the grades are distributed. Therefore, we created two indexes based on subject, cNumber. However, these two indexes both did not perform well. Then, we created another two indexes on multi columns. We created a combination of subject and cNumber, and another index with instructor first name and last name. The result turns out the index that uses instructor names has the best performance.