

Database Implementation

Connection into GCP via Terminal

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to cryptic-heaven-390921.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
maxbhartman@cloudshell:~ (cryptic-heaven-390921)$  
maxbhartman@cloudshell:~ (cryptic-heaven-390921)$ gcloud sql connect cs-411-test --user=root --quiet  
Allowlisting your IP for incoming connection for 5 minutes...done.  
Connecting to database with SQL user [root].Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 29706  
Server version: 8.0.31-google (Google)  
  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> █
```

DDL Commands to create tables in Data

```
CREATE DATABASE IF NOT EXISTS WSPG_DB;
USE WSPG_DB;

CREATE TABLE Department (
    DepartmentName VARCHAR(255) PRIMARY KEY,
    Enrollment INT,
    CourseCount INT
    FemalePct float
);

CREATE TABLE CourseInfo (
    CourseCode VARCHAR(10) NOT NULL,
    ProfessorName VARCHAR(255) NOT NULL,
    CourseTitle VARCHAR(255),
    CourseDescription TEXT,
    Credits INT,
    DepartmentName VARCHAR(255),
    PRIMARY KEY (CourseCode, ProfessorName),
    FOREIGN KEY (DepartmentName) REFERENCES Department(DepartmentName) ON DELETE CASCADE
);

CREATE TABLE ProfessorInfo (
    ProfessorName VARCHAR(255) PRIMARY KEY,
    Number5s INT,
    Number4s INT,
    Number3s INT,
    Number2s INT,
    Number1s INT
);

CREATE TABLE AverageGPA (
    CourseCode VARCHAR(10) NOT NULL,
    ProfessorName VARCHAR(255) NOT NULL,
    NumberAPlus INT,
    NumberA INT,
    NumberAMinus INT,
    NumberBPlus INT,
    NumberB INT,
    NumberBMinus INT,
    NumberCPlus INT,
    NumberC INT,
    NumberCMinus INT,
    NumberDPlus INT,
    NumberD INT,
    NumberDMinus INT,
    NumberF INT,
    NumberW INT,
    PRIMARY KEY (CourseCode, ProfessorName),
    FOREIGN KEY (CourseCode) REFERENCES CourseInfo(CourseCode),
    FOREIGN KEY (ProfessorName) REFERENCES ProfessorInfo(ProfessorName)
);

CREATE TABLE Awards (
    ProfessorName VARCHAR(255) NOT NULL,
    Award VARCHAR(255) NOT NULL,
    Department VARCHAR(255),
    CourseCode VARCHAR(10) NOT NULL,
    Term VARCHAR(10),
    PRIMARY KEY (ProfessorName, Term),
    FOREIGN KEY (ProfessorName) REFERENCES ProfessorInfo(ProfessorName),
    FOREIGN KEY (CourseCode) REFERENCES CourseInfo(CourseCode)
);

CREATE TABLE Users (
    NetID VARCHAR(255) PRIMARY KEY,
    Major VARCHAR(255),
    GraduatingYear INT
);

CREATE TABLE MatchResult (
    NetID VARCHAR(255) NOT NULL,
    CourseCode VARCHAR(10) NOT NULL,
    Response INT,
    PRIMARY KEY (NetID, CourseCode),
    FOREIGN KEY (NetID) REFERENCES Users(NetID),
    FOREIGN KEY (CourseCode) REFERENCES CourseInfo(CourseCode)
);
```

Three tables with over 1000 rows

```
mysql> select count(*) from CourseInfo;
+-----+
| count(*) |
+-----+
|      5050 |
+-----+
1 row in set (0.01 sec)
```

```
mysql> select count(*) from AverageGPA;
+-----+
| count(*) |
+-----+
|     11732 |
+-----+
1 row in set (0.01 sec)
```

```
mysql> select count(*) from ProfessorInfo;
+-----+
| count(*) |
+-----+
|     9308 |
+-----+
1 row in set (0.00 sec)
```

Queries

Average Department GPA Top 15 Results

DepartmentName	AvgGPA
EPOL	3.94
ERAM	3.94
MUSC	3.93
MBA	3.88
BUS	3.87
VCM	3.85
CHP	3.84
REES	3.84
SPED	3.83
TE	3.83
CI	3.81
IS	3.81
LAS	3.80
REHB	3.79
AHS	3.77

15 rows in set (0.09 sec)

This query computes the Average GPA for courses in each department. This query **joins** the Department, CourseInfo, and AverageGPA tables together. The query uses **Group by** to group the output AverageGPA by DepartmentName. This query will be used in our final Web App, so we can display the average gpa for the department of the class the user is prompted with.

Courses with Average GPA > 3.2 and 30% women in that Course's Department

DepartmentName	AvgGPA	FemalePct
ERAM	3.94	0.57
MUSC	3.93	0.59
MBA	3.88	0.49
BUS	3.87	0.89
VCM	3.85	0.55
REES	3.84	0.68
CHP	3.84	0.48
SPED	3.83	0.31
CI	3.81	0.67
IS	3.81	0.4
LAS	3.80	0.37
REHB	3.79	0.77
ALEC	3.77	0.57
KOR	3.76	0.63
SOCW	3.75	0.72
BDI	3.75	0.65
TRST	3.75	0.4
ENSU	3.74	0.42

This query finds departments with GPAs above 3.2 in a department that has at least 30% women enrolled. This query **joins** the Department, CourseInfo, and AverageGPA tables together. The query uses **Group by** to group the output AverageGPA by DepartmentName. This query will be used in our final Web App, so we can find departments with a high GPA (3.2+) and high diversity (30%+ women). This will be used so female users know which departments support diversity, equity, and inclusion.

Average GPA for each course

CourseCode	AvgGPA
MUSC484	3.99
IS534	3.99
MUSC480	3.98
TE510	3.98
MUSC476	3.98
FIN394	3.98
ACCY518	3.97
ENG411	3.97
ABE199	3.97
CI443	3.97
KIN449	3.97
FSHN249	3.97
IS544	3.97
IS565	3.97
ENG377	3.97

This query lists courses in order of GPA in the course. This query **joins** CourseInfo and AverageGPA tables together. The query uses **Group by** to group the output AverageGPA by CourseCode. This query is essential to our final app as this is how we will display the average gpa for a course presented to the user.

Great Professors: 3.5 Average GPA in courses taught and won awards

ProfessorName	AvgGPA
Hensley, M	3.95
Chen, M	3.95
Grift, T	3.94
Leiby, J	3.93
Craig, A	3.93
Tate, A	3.93
Parsons, C	3.92
Escobar-Sawicki, C	3.92
Napolitano, C	3.91
Rangarajan, A	3.90
Smith, N	3.90
Light Shriner, C	3.90
Mendes, C	3.89
Zarate, K	3.89
Erickson, K	3.88

This query distinguishes the top faculty at this university by filtering professors by the average gpa in their courses and if they won any awards. This query **joins** CourseInfo, AverageGPA, and Professor tables together. The query uses **Group by** to group the output AverageGPA by ProfessorName. This query is essential to our final app as this is how we will display if the course a user is prompted for has a great professor teaching the course.

Indexing

Query 1 (Course_AvgGPA.sql)

Baseline

- Nested loop left join cost: 5066.99

```
| -> Sort: AvgGPA DESC (actual_time=109.598..109.747 rows=3059 loops=1)
    -> Stream results (cost=8350.61 rows=32836) (actual_time=0.231..108.431 rows=3059 loops=1)
        -> Group aggregate: avgif(((((((((ag.NumberAPlus + ag.NumberAMinus) * ag.NumberBPlus) + ag.NumberB) * ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberBPPlus) +
        ag.NumberDPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67)) +
        + (ag.NumberCPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67))
        -> Nested loop left join (cost=5066.99 rows=32836) (actual_time=0.101..44.075 rows=43837 loops=1)
            -> Covering index scan on c using PRIMARY (cost=542.85 rows=4866) (actual_time=0.067..2.148 rows=5050 loops=1)
            -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=7) (actual_time=0.004..0.008 rows=8 loops=5050)
|
```

Experiment 1

CREATE INDEX counts_idx on AverageGPA(NumberAPlus, NumberA, NumberAMinus, NumberBPlus, NumberB, NumberBMinus, NumberCPlus, NumberC, NumberCMinus, NumberDPlus, NumberD, NumberDMinus, NumberF);

- Nested loop left join cost: 5066.99
- Compared to the baseline, this is the same.

```
| -> Sort: AvgGPA DESC (actual_time=108.587..108.722 rows=3059 loops=1)
    -> Stream results (cost=8350.61 rows=32836) (actual_time=0.235..107.490 rows=3059 loops=1)
        -> Group aggregate: avgif(((((((((ag.NumberAPlus + ag.NumberAMinus) * ag.NumberBPlus) + ag.NumberB) * ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberBPPlus) +
        ag.NumberDPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67)) +
        + (ag.NumberCPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67))
        -> Nested loop left join (cost=5066.99 rows=32836) (actual_time=0.066..45.838 rows=43837 loops=1)
            -> Covering index scan on c using PRIMARY (cost=542.85 rows=4866) (actual_time=0.067..2.071 rows=5050 loops=1)
            -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=7) (actual_time=0.004..0.008 rows=8 loops=5050)
|
```

Experiment 2

CREATE INDEX CourseCode_idx on AverageGPA(CourseCode);

- Nested loop left join: cost = 5066.99
- Compared to the baseline, this is the same.

```
| -> Sort: AvgGPA DESC (actual_time=107.457..107.625 rows=3059 loops=1)
    -> Stream results (cost=8350.61 rows=32836) (actual_time=0.246..106.454 rows=3059 loops=1)
        -> Group aggregate: avgif(((((((((ag.NumberAPlus + ag.NumberAMinus) * ag.NumberBPlus) + ag.NumberB) * ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberBPPlus) +
        ag.NumberDPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67)) +
        + (ag.NumberCPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67))
        -> Nested loop left join (cost=5066.99 rows=32836) (actual_time=0.122..43.919 rows=43837 loops=1)
            -> Covering index scan on PRIMARY (CourseCode=c.CourseCode) (cost=542.85 rows=4866) (actual_time=0.060..2.141 rows=5050 loops=1)
            -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=7) (actual_time=0.003..0.008 rows=8 loops=5050)
|
```

Experiment 3

CREATE INDEX CourseCode_idx on CourseInfo(CourseCode);

- Nested loop left join: cost = 0.70
- Compared to the baseline, the cost is lower so we have a performance gain

```
| -> Sort: AvgGPA DESC (actual_time=0.020..0.020 rows=0 loops=1)
    -> Stream results (cost=8350.61 rows=32836) (actual_time=0.014..0.014 rows=0 loops=1)
        -> Group aggregate: avgif((((((((((ag.NumberAPlus + ag.NumberAMinus) * ag.NumberBPlus) + ag.NumberB) * ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberBPPlus) +
        ag.NumberDPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67)) +
        + (ag.NumberCPlus * 2.33)) + (ag.NumberCPlus * 2.0) + (ag.NumberCMinus * 1.67)) + (ag.NumberBPlus * 1.33)) * (ag.NumberB * 4.0) + (ag.NumberBMinus * 3.67) + (ag.NumberBPlus * 3.33)) * (ag.NumberB * 3.0) + (ag.NumberBMinus * 2.67))
        -> Nested loop left join (cost=0.70 rows=1) (actual_time=0.012..0.012 rows=0 loops=1)
            -> Covering index scan on c using CourseCode_idx (cost=0.35 rows=1) (actual_time=0.011..0.011 rows=0 loops=1)
            -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.35 rows=1) (never executed)
|
```

Final index design:

For this query, we will be using our experiment 3 (CREATE INDEX CourseCode_idx on CourseInfo(CourseCode);) as our way of indexing. The reason we are choosing this is because

our cost of 0.70 for the nested loop left join is significantly lower than our baseline and other 3 experiments, which were all 5066.99.

Query 2 (Diversity.sql):

Baseline

- Nested loop inner join: 50.00

```
|--> Sort: g.AvgGPA DESC, d.FemalePct DESC (actual time=2.703..2.703 rows=0 loops=1)
--> Stream results (cost=50.00 rows=0) (actual time=1.866..1.866 rows=0 loops=1)
--> Nested loop inner join (cost=50.00 rows=0) (actual time=1.866..1.866 rows=0 loops=1)
--> Table scan on g (cost=2.50..2.50 rows=0) (actual time=1.861..1.861 rows=0 loops=1)
--> > Materialize CTE DeptGPA (cost=0..0.00 rows=0) (actual time=1.860..1.860 rows=0 loops=1)
--> > Filter: (roundavg(if(((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberBMinus) * ag.NumberCPlus) * ag.NumberCMinus) + ag.NumberDPlus) * ag.NumberDMinus) + ag.NumberEPlus) * ag.NumberEMinus) + ag.NumberFPlus) * ag.NumberFMinus) < 3.67) / ((ag.NumberBPlus * 3.33) + (ag.NumberB * 3.0) + (ag.NumberCPlus * 2.67) + (ag.NumberC * 2.0) + (ag.NumberDPlus * 1.33) + (ag.NumberD * 1.0) + (ag.NumberEPlus * 0.67) + (ag.NumberE * 0) / (((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberB) * ag.NumberCPlus) * ag.NumberC) * ag.NumberDPlus) * ag.NumberD) + ag.NumberEPlus) * ag.NumberE) + ag.NumberFPlus) * ag.NumberF) ) > 3.2) (actual time=1.782..1.782 rows=0 loops=1)
--> > Aggregate using temporary table (actual time=0.960..0.967 rows=190 loops=1)
--> > Left hash join (ag.CourseCode = ci.CourseCode) (cost=19.63 rows=190) (actual time=0..310..0.632 rows=190 loops=1)
--> > Nested loop left join (cost=85.75 rows=190) (actual time=0..180..0.464 rows=190 loops=1)
--> > > Covering index scan on d using PRIMARY (cost=19.25 rows=190) (actual time=0..158..0.223 rows=190 loops=1)
--> > > Covering index lookup on ci using DepartmentName (DepartmentName=d.DepartmentName) (cost=0..25 rows=1) (actual time=0..001..0.001 rows=1 loops=190)
--> Hash
--> Table scan on ag (cost=0..00 rows=1) (actual time=0..036..0.036 rows=0 loops=1)
--> > Filter: (d.FemalePct > 0.3) (cost=0..25 rows=0..3) (never executed)
--> > Single-row index lookup on d using PRIMARY (DepartmentName=d.DepartmentName) (cost=0..25 rows=1) (never executed)
```

Experiment 1

CREATE INDEX woman_idx on Department(FemalePct);

- Nested loop inner join: cost = 6033.61
- Compared to the baseline, this is a higher cost so we have a performance degradation

```
|--> Sort: g.AvgGPA DESC, d.FemalePct DESC (actual time=102.120..102.125 rows=91 loops=1)
--> Stream results (cost=6033.61 rows=467) (actual time=102.180..102.182 rows=91 loops=1)
--> Nested loop inner join (cost=6033.61 rows=467) (actual time=101.845..101.849 rows=91 loops=1)
--> > Table scan on g (cost=4015.35..4087.92 rows=5559) (actual time=101.861..101.881 rows=116 loops=1)
--> > > Materialize CTE DeptGPA (cost=4015.35..4015.93 rows=5559) (actual time=101.861..101.861 rows=116 loops=1)
--> > > Filter: (roundavg(if(((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberBMinus) * ag.NumberCPlus) * ag.NumberCMinus) + ag.NumberDPlus) * ag.NumberDMinus) + ag.NumberEPlus) * ag.NumberEMinus) + ag.NumberFPlus) * ag.NumberFMinus) < 3.67) / ((ag.NumberBPlus * 3.33) + (ag.NumberB * 3.0) + (ag.NumberCPlus * 2.67) + (ag.NumberC * 2.0) + (ag.NumberDPlus * 1.33) + (ag.NumberD * 1.0) + (ag.NumberEPlus * 0.67) + (ag.NumberE * 0) / (((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberB) * ag.NumberCPlus) * ag.NumberC) * ag.NumberDPlus) * ag.NumberD) + ag.NumberEPlus) * ag.NumberE) + ag.NumberFPlus) * ag.NumberF) ) > 3.2) (actual time=0.232..101.735 rows=116 loops=1)
--> > Group aggregate: avg(if(((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberBMinus) * ag.NumberCPlus) * ag.NumberCMinus) + ag.NumberDPlus) * ag.NumberDMinus) + ag.NumberEPlus) * ag.NumberEMinus) + ag.NumberFPlus) * ag.NumberFMinus) < 3.67) / ((ag.NumberBPlus * 3.33) + (ag.NumberB * 3.0) + (ag.NumberCPlus * 2.67) + (ag.NumberC * 2.0) + (ag.NumberDPlus * 1.33) + (ag.NumberD * 1.0) + (ag.NumberEPlus * 0.67) + (ag.NumberE * 0) / (((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberB) * ag.NumberCPlus) * ag.NumberC) * ag.NumberDPlus) * ag.NumberD) + ag.NumberEPlus) * ag.NumberE) + ag.NumberFPlus) * ag.NumberF) ) > 3.2) (actual time=0.232..101.638 rows=190 loops=1)
--> > Nested loop left join (cost=858.27 rows=5559) (actual time=0..156..0.556..0.280 rows=5063 loops=1)
--> > > Covering index scan on d using PRIMARY (cost=19.25 rows=190) (actual time=0..035..0.113 rows=190 loops=1)
--> > > Covering index lookup on ci using DepartmentName (DepartmentName=d.DepartmentName) (cost=2..03 rows=29) (actual time=0..005..0.010 rows=27 loops=190)
--> > > Index lookup on ag using PRIMARY (CourseCode=ci.CourseCode) (cost=0..25 rows=1) (actual time=0..003..0.007 rows=8 loops=5063)
--> > Filter: (d.FemalePct > 0..3) (cost=0..25 rows=0..3) (actual time=0..001..0.001 rows=1 loops=16)
--> > Single-row index lookup on d using PRIMARY (DepartmentName=d.DepartmentName) (cost=0..25 rows=1) (actual time=0..001..0.001 rows=1 loops=16)
```

Experiment 2

CREATE INDEX counts_idx on AverageGPA(NumberAPlus, NumberA, NumberAMinus, NumberBPlus, NumberB, NumberBMinus, NumberCPlus, NumberC, NumberCMinus, NumberDPlus, NumberD, NumberDMinus, NumberF);

- Nested loop inner join: cost = 33396.53
- Compared to the baseline, this is a higher cost (even higher than experiment 1) so this is also a performance degradation.

```
|--> Sort: g.AvgGPA DESC, d.FemalePct DESC (actual time=101.950..101.955 rows=91 loops=1)
--> Stream results (cost=33396.53 rows=15601) (actual time=101.695..101.897 rows=91 loops=1)
--> Nested loop inner join (cost=33396.53 rows=15601) (actual time=101.689..101.863 rows=91 loops=1)
--> > Table scan on g (cost=1626.07..16426.07 rows=46809) (actual time=101.677..101.655 rows=116 loops=1)
--> > > Materialize CTE DeptGPA (cost=16426.07..16426.07 rows=46809) (actual time=101.655..101.655 rows=116 loops=1)
--> > > Filter: (roundavg(if(((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberBMinus) * ag.NumberCPlus) * ag.NumberCMinus) + ag.NumberDPlus) * ag.NumberDMinus) + ag.NumberEPlus) * ag.NumberEMinus) + ag.NumberFPlus) * ag.NumberFMinus) < 3.67) / ((ag.NumberBPlus * 3.33) + (ag.NumberB * 3.0) + (ag.NumberCPlus * 2.67) + (ag.NumberC * 2.0) + (ag.NumberDPlus * 1.33) + (ag.NumberD * 1.0) + (ag.NumberEPlus * 0.67) + (ag.NumberE * 0) / (((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberB) * ag.NumberCPlus) * ag.NumberC) * ag.NumberDPlus) * ag.NumberD) + ag.NumberEPlus) * ag.NumberE) + ag.NumberFPlus) * ag.NumberF) ) > 3.2) (actual time=0.223..101.446 rows=190 loops=1)
--> > Group aggregate: avg(if(((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberBMinus) * ag.NumberCPlus) * ag.NumberCMinus) + ag.NumberDPlus) * ag.NumberDMinus) + ag.NumberEPlus) * ag.NumberEMinus) + ag.NumberFPlus) * ag.NumberFMinus) < 3.67) / ((ag.NumberBPlus * 3.33) + (ag.NumberB * 3.0) + (ag.NumberCPlus * 2.67) + (ag.NumberC * 2.0) + (ag.NumberDPlus * 1.33) + (ag.NumberD * 1.0) + (ag.NumberEPlus * 0.67) + (ag.NumberE * 0) / (((((((((ag.NumberAPlus + ag.NumberA) * ag.NumberAMinus) + ag.NumberBPlus) * ag.NumberB) * ag.NumberCPlus) * ag.NumberC) * ag.NumberDPlus) * ag.NumberD) + ag.NumberEPlus) * ag.NumberE) + ag.NumberFPlus) * ag.NumberF) ) > 3.2) (actual time=0.223..101.446 rows=190 loops=1)
--> Nested loop left join (cost=7064.35 rows=46809) (actual time=0..078..42.352 rows=4355 loops=1)
--> > Nested loop left join (cost=7064.35 rows=46809) (actual time=0..078..42.352 rows=4355 loops=1)
--> > > Covering index scan on d using PRIMARY (cost=19.25 rows=190) (actual time=0..029..0.107 rows=190 loops=1)
--> > > Covering index lookup on ci using DepartmentName (DepartmentName=d.DepartmentName) (cost=2..03 rows=29) (actual time=0..005..0.010 rows=27 loops=190)
--> > > Index lookup on ag using PRIMARY (CourseCode=ci.CourseCode) (cost=0..26 rows=8) (actual time=0..004..0.007 rows=8 loops=5063)
--> > Filter: (d.FemalePct > 0..3) (cost=0..25 rows=0..3) (actual time=0..001..0.001 rows=1 loops=16)
--> > Single-row index lookup on d using PRIMARY (DepartmentName=d.DepartmentName) (cost=0..25 rows=1) (actual time=0..001..0.001 rows=1 loops=16)
```

Experiment 3

```
CREATE INDEX woman_idx on Department(DepartmentName);
```

- Nested loop inner join: cost = 33396.53
 - Compared to the baseline, this is a higher cost (higher than experiment 1 and same as experiment 2) so this is a performance degradation

Final index design:

For this query, we will be using our baseline without indexing. The reason is because for all 3 of our experiments, our cost increased so it would make more sense for us to stick with no indexing as it led us to having the lowest cost.

Query 3 (Average GPA for Dept.)

BASELINE

- Nested loop left join: cost = 7064.35

```
-----+
| -> Group aggregate: avg(if(((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberAMinus) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|   + ag.NumberDMinus) + ag.NumberF) = 0),NULL,((((((((((ag.NumberAPlus * 4.0) + (ag.NumberA * 4.0)) + (ag.NumberAMinus * 3.67)) + (ag.NumberBPlus * 3.33)) + (ag.NumberB * 3.0)) + (ag.NumberBMinus * 2.67)) + (ag.NumberCPlus * 2.33)) + (ag.NumberC * 2.0)) + (ag.NumberCMinus * 1.67)) + (ag.NumberDPlus * 1.33)) + (ag.NumberD * 1.0)) + (ag.NumberDMinus * 0.67)) + (ag.NumberF * 0)) / (((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|   Line=0.318..103.149 rows=190 loops=1
| -> Nested loop left join (cost=7064.35 rows=46809) (actual time=0.147..43.321 rows=43554 loops=1)
|   -> Nested loop left join (cost=958.27 rows=5559) (actual time=0.069..2.448 rows=5063 loops=1)
|       -> Covering index scan on ag using PRIMARY (DepartmentName=d.DepartmentName) (cost=2.03 rows=29) (actual time=0.005..0.010 rows=27 loops=190)
|       -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=8) (actual time=0.004..0.007 rows=8 loops=5063)
|
```

Experiment 1

CREATE INDEX woman_idx on Department(DepartmentName);

- Nested loop left join: cost = 7064.35
- Compared to the baseline, this is the same

```
-----+
| -> Sort: AvgGPA DESC (actual time=102.428..102.440 rows=190 loops=1)
|   -> Stream results (cost=11745.21 rows=6809) (actual time=102.21..102.291 rows=190 loops=1)
|       -> Group aggregate: avg(if((((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberAMinus) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|         + ag.NumberDMinus) + ag.NumberF) = 0),NULL,((((((((((ag.NumberAPlus * 4.0) + (ag.NumberA * 4.0)) + (ag.NumberAMinus * 3.67)) + (ag.NumberBPlus * 3.33)) + (ag.NumberB * 3.0)) + (ag.NumberBMinus * 2.67)) + (ag.NumberCPlus * 2.33)) + (ag.NumberC * 2.0)) + (ag.NumberCMinus * 1.67)) + (ag.NumberDPlus * 1.33)) + (ag.NumberD * 1.0)) + (ag.NumberDMinus * 0.67)) + (ag.NumberF * 0)) / (((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|         Line=0.208..102.153 rows=190 loops=1
| -> Nested loop left join (cost=7064.35 rows=46809) (actual time=0.064..42.898 rows=43554 loops=1)
|   -> Nested loop left join (cost=958.27 rows=5559) (actual time=0.044..2.422 rows=5063 loops=1)
|       -> Covering index scan on d using woman_idx (DepartmentName=d.DepartmentName) (cost=19.25 rows=190) (actual time=0.026..0.122 rows=90 loops=1)
|       -> Covering index lookup on cl using DepartmentName (DepartmentName=d.DepartmentName) (cost=2.03 rows=29) (actual time=0.005..0.010 rows=27 loops=190)
|       -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=8) (actual time=0.004..0.007 rows=8 loops=5063)
|
```

Experiment 2

CREATE INDEX woman_idx on CourseInfo(CourseCode);

- Nested loop left join: cost = 7064.35
- Compared to the baseline, this is the same

```
-----+
| -> Sort: AvgGPA DESC (actual time=103.328..103.336 rows=190 loops=1)
|   -> Stream results (cost=11745.21 rows=6809) (actual time=103.199 rows=190 loops=1)
|       -> Group aggregate: avg(if((((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberAMinus) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|         + ag.NumberDMinus) + ag.NumberF) = 0),NULL,((((((((((ag.NumberAPlus * 4.0) + (ag.NumberA * 4.0)) + (ag.NumberAMinus * 3.67)) + (ag.NumberBPlus * 3.33)) + (ag.NumberB * 3.0)) + (ag.NumberBMinus * 2.67)) + (ag.NumberCPlus * 2.33)) + (ag.NumberC * 2.0)) + (ag.NumberCMinus * 1.67)) + (ag.NumberDPlus * 1.33)) + (ag.NumberD * 1.0)) + (ag.NumberDMinus * 0.67)) + (ag.NumberF * 0)) / (((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|         Line=0.218..103.041 rows=190 loops=1
| -> Nested loop left join (cost=7064.35 rows=46809) (actual time=0.072..42.807 rows=43554 loops=1)
|   -> Nested loop left join (cost=958.27 rows=5559) (actual time=0.053..2.323 rows=5063 loops=1)
|       -> Covering index scan on d using PRIMARY (cost=19.25 rows=190) (actual time=0.033..0.104 rows=90 loops=1)
|       -> Covering index lookup on ci using DepartmentName (DepartmentName=d.DepartmentName) (cost=2.03 rows=29) (actual time=0.005..0.010 rows=27 loops=190)
|       -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=8) (actual time=0.004..0.007 rows=8 loops=5063)
|
```

Experiment 3

CREATE INDEX counts_idx on AverageGPA(NumberAPlus, NumberA, NumberAMinus, NumberBPlus, NumberB, NumberBMinus, NumberCPlus, NumberC, NumberCMinus, NumberDPlus, NumberD, NumberDMinus, NumberF);

- Nested loop left join: cost = 7064.35
- Compared to the baseline, this is the same

```
-----+
| -> Sort: AvgGPA DESC (actual time=103.636..103.644 rows=190 loops=1)
|   -> Stream results (cost=11745.21 rows=6809) (actual time=103.504 rows=190 loops=1)
|       -> Group aggregate: avg(if((((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberAMinus) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|         + ag.NumberDMinus) + ag.NumberF) = 0),NULL,((((((((((ag.NumberAPlus * 4.0) + (ag.NumberA * 4.0)) + (ag.NumberAMinus * 3.67)) + (ag.NumberBPlus * 3.33)) + (ag.NumberB * 3.0)) + (ag.NumberBMinus * 2.67)) + (ag.NumberCPlus * 2.33)) + (ag.NumberC * 2.0)) + (ag.NumberCMinus * 1.67)) + (ag.NumberDPlus * 1.33)) + (ag.NumberD * 1.0)) + (ag.NumberDMinus * 0.67)) + (ag.NumberF * 0)) / (((((((((ag.NumberAPlus + ag.NumberA) + ag.NumberBPlus) + ag.NumberB) + ag.NumberBMinus) + ag.NumberCPlus) + ag.NumberC) + ag.NumberCMinus) + ag.NumberDPlus) + ag.NumberD)
|         Line=0.356..103.357 rows=190 loops=1
| -> Nested loop left join (cost=7064.35 rows=46809) (actual time=0.096..43.252 rows=43554 loops=1)
|   -> Nested loop left join (cost=958.27 rows=5559) (actual time=0.074..2.438 rows=5063 loops=1)
|       -> Covering index scan on d using PRIMARY (cost=19.25 rows=190) (actual time=0.047..0.122 rows=90 loops=1)
|       -> Covering index lookup on cl using DepartmentName (DepartmentName=d.DepartmentName) (cost=2.03 rows=29) (actual time=0.005..0.010 rows=27 loops=190)
|       -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=8) (actual time=0.004..0.007 rows=8 loops=5063)
|
```

Final index design:

For this query, we will be using our baseline without indexing. The reason is because we did not find any difference in cost when we tried adding different indices to different attributes. We think this is the case, because of the cardinality of the Department table. Changing the query index did not impact the cost, since the table is small enough where the SQL engine likely just computes a one-pass join.

Query 4 (Professor)

Baseline

- Nested loop left join: cost = 2753888.18

```
|----+  
| -> Sort: AvgGPA DESC (actual time=32.936..32.944 rows=195 loops=1)  
|   -> Filter: (AvgGPA > 3.5) (actual time=32.632..32.847 rows=195 loops=1)  
|     -> Table scan on <temporary> (actual time=32.621..32.740 rows=451 loops=1)  
|       -> Aggregate using temporary table (actual time=32.619..32.619 rows=451 loops=1)  
|         -> Nested loop left join (cost=2753888.18 rows=24593728) (actual time=0.719..13.286 rows=7903 loops=1)  
|           -> Nested loop inner join (cost=293187.65 rows=2920956) (actual time=0.682..4.462 rows=1084 loops=1)  
|             -> Covering index scan on c using DepartmentName (cost=574.15 rows=5179) (actual time=0.031..0.324 rows=5050 loops=1)  
|               -> Single-row index lookup on <subquery2> using <auto_distinct_key> (ProfessorName=c.ProfessorName) (actual time=0.001..0.001 rows=0 loops=5050)  
|                 -> Materialize with deduplication (cost=114.05..114.05 rows=564) (actual time=0.646..0.646 rows=451 loops=1)  
|                   -> Covering index scan on Awards using CourseCode (cost=57.65 rows=564) (actual time=0.088..0.208 rows=564 loops=1)  
|             -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=8) (actual time=0.004..0.008 rows=7 loops=1084)  
|
```

EXPERIMENT 1

CREATE INDEX counts_idx on AverageGPA(NumberAPlus, NumberA, NumberAMinus, NumberBPlus, NumberB, NumberBMinus, NumberCPlus, NumberC, NumberCMINus, NumberDPlus, NumberD, NumberDMINus, NumberF);

- Nested loop left join: cost = 2753888.18
- Compared to the baseline, this is the same

```
|----+  
| -> Sort: AvgGPA DESC (actual time=28.083..28.095 rows=195 loops=1)  
|   -> Filter: (AvgGPA > 3.5) (actual time=27.814..27.999 rows=195 loops=1)  
|     -> Table scan on <temporary> (actual time=27.806..27.908 rows=451 loops=1)  
|       -> Aggregate using temporary table (actual time=27.804..27.804 rows=451 loops=1)  
|         -> Nested loop left join (cost=2753888.18 rows=24593728) (actual time=0.432..12.344 rows=7903 loops=1)  
|           -> Nested loop inner join (cost=293187.65 rows=2920956) (actual time=0.402..4.093 rows=1084 loops=1)  
|             -> Covering index scan on c using DepartmentName (cost=574.15 rows=5179) (actual time=0.044..0.917 rows=5050 loops=1)  
|               -> Single-row index lookup on <subquery2> using <auto_distinct_key> (ProfessorName=c.ProfessorName) (actual time=0.001..0.001 rows=0 loops=5050)  
|                 -> Materialize with deduplication (cost=114.05..114.05 rows=564) (actual time=0.354..0.354 rows=451 loops=1)  
|                   -> Covering index scan on Awards using CourseCode (cost=57.65 rows=564) (actual time=0.019..0.190 rows=564 loops=1)  
|             -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0.26 rows=8) (actual time=0.004..0.007 rows=7 loops=1084)  
|
```

EXPERIMENT 2

CREATE INDEX counts_idx on CourseInfo(ProfessorName)

- Nested loop left join: cost = 1663.66
- Compared to the baseline, this has a significantly lower cost so we have a performance gain

```
|----+  
| -> Sort: AvgGPA DESC (actual time=78.489..78.503 rows=195 loops=1)  
|   -> Filter: (AvgGPA > 3.5) (actual time=78.211..78.402 rows=195 loops=1)  
|     -> Table scan on <temporary> (actual time=78.203..78.312 rows=451 loops=1)  
|       -> Aggregate using temporary table (actual time=78.201..78.201 rows=451 loops=1)  
|         -> Nested loop left join (cost=1663.66 rows=8642) (actual time=1..704..61.959 rows=7903 loops=1)  
|           -> Nested loop inner join (cost=799.04 rows=1026) (actual time=1..680..51.709 rows=1084 loops=1)  
|             -> Remove duplicates from input sorted on PRIMARY (cost=80..54 rows=564) (actual time=0..587..48.200 rows=451 loops=1)  
|               -> Covering index scan on Awards using PRIMARY (cost=80..54 rows=564) (actual time=0..586..48.076 rows=564 loops=1)  
|                 -> Covering index lookup on c using woman_idx (ProfessorName=Awards.ProfessorName) (cost=616.06 rows=2) (actual time=0..004..0.005 rows=2 loops=451)  
|             -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode) (cost=0..72 rows=8) (actual time=0..005..0.009 rows=7 loops=1084)  
|
```

EXPERIMENT 3

CREATE INDEX counts_idx on Awards(ProfessorName)

- Nested loop left join: cost = 2753888.18
- Compared to the baseline, this is the same

```

+-----+
| -> Sort: AvgGPA DESC  (actual time=30.010..30.021 rows=195 loops=1)
  -> Filter: (AvgGPA > 3.5)  (actual time=29.720..29.919 rows=195 loops=1)
    -> Table scan on <temporary>  (actual time=29.710..29.827 rows=451 loops=1)
      -> Aggregate using temporary table  (actual time=29.708..29.708 rows=451 loops=1)
        -> Nested loop left join  (cost=2753888.18 rows=24593728) (actual time=0.395..13.435 rows=7903 loops=1)
          -> Nested loop inner join  (cost=293187.65 rows=2920956) (actual time=0.372..4.440 rows=1084 loops=1)
            -> Covering index scan on c using DepartmentName  (cost=574.15 rows=5179) (actual time=0.036..1.013 rows=5050 loops=1)
            -> Single-row index lookup on <subquery> using <auto_distinct key> (ProfessorName=c.ProfessorName)  (actual time=0.001..0.001 rows=0 loops=1)
              -> Materialize with deduplication  (cost=114.05..114.05 rows=564) (actual time=0.332..0.332 rows=451 loops=1)
                -> Covering index scan on Awards using CourseCode  (cost=57.65 rows=564) (actual time=0.029..0.159 rows=564 loops=1)
                  -> Index lookup on ag using PRIMARY (CourseCode=c.CourseCode)  (cost=0.26 rows=8) (actual time=0.004..0.008 rows=7 loops=1084)
|
+-----+

```

Final index design:

For this query, we will be using our experiment 2 (CREATE INDEX counts_idx on CoursesInfo(ProfessorName)) to index. The reason is because our cost of 1663.66 for the nested loop left join is significantly lower than our cost for the baseline and other two experiments, which were all 275388.18.