

**Project Phase II: University Database System**

SOFE3700U:Data Management Systems

Group 13

Nov. 7, 2021

Rayyan Mohammed(100752351)

Daniyal Khan (100750029)

Muhammad Zaeem Khalid (100746801)

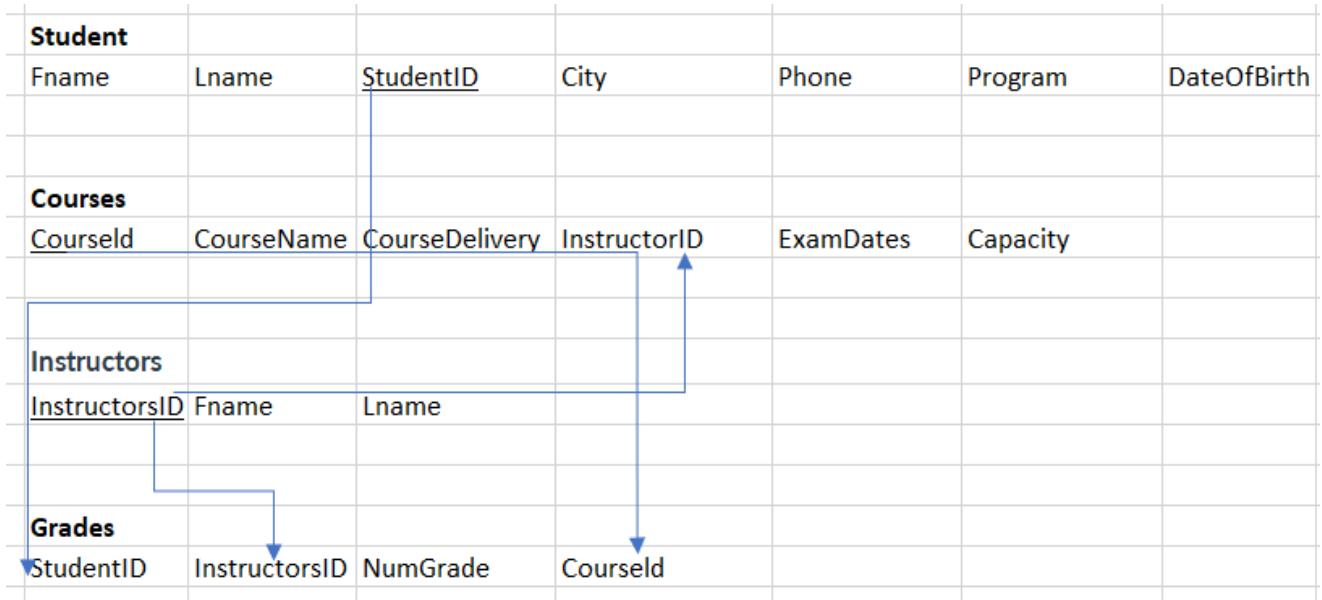
Yousif Sarmad (100746887)

Charles Olagunju (100749818)

Deliverables:

•Part A: Relational Schema (5points)

- Create relational database schema for the proposed application area from Phase I:



- Actual SQL create table commands are required.

```
Algorithm: Default Lock Type: Default
1 CREATE SCHEMA `FinalProject` ;
2
```

Creates a schema called FinalProject

```
Algorithm: Default Lock Type: Default
1   ⊖ CREATE TABLE `FinalProject`.`Student` (
2       `StudentID` INT NOT NULL,
3       `Fname` VARCHAR(45) NOT NULL,
4       `Lname` VARCHAR(45) NOT NULL,
5       `City` VARCHAR(45) NOT NULL,
6       `Phone` INT NOT NULL,
7       `DateOfBirth` DATETIME NOT NULL,
8       `Program` VARCHAR(45) NOT NULL,
9       PRIMARY KEY (`StudentID`));
10
```

SQL create table command for Student table

```
1   ⊖ CREATE TABLE `FinalProject`.`Instructors` (
2       `InstructorID` INT NOT NULL,
3       `Fname` VARCHAR(45) NOT NULL,
4       `Lname` VARCHAR(45) NOT NULL,
5       PRIMARY KEY (`InstructorID`));
6
```

SQL create table command for Instructor table

```
Algorithm: Default Lock Type: Default
1   ⊖ CREATE TABLE `FinalProject`.`Courses` (
2       `CourseID` INT NOT NULL,
3       `CourseName` VARCHAR(45) NOT NULL,
4       `CourseDelivery` VARCHAR(45) NOT NULL,
5       `Capacity` INT NOT NULL,
6       `ExamDates` DATETIME NOT NULL,
7       `InstructorID` INT NOT NULL,
8       PRIMARY KEY (`CourseID`));
9
```

SQL create table command for Courses table

```
1  ALTER TABLE `FinalProject`.`Courses`  
2  ADD INDEX `InstructorID_idx` (`InstructorID` ASC) VISIBLE;  
3  ;  
4  ALTER TABLE `FinalProject`.`Courses`  
5  ADD CONSTRAINT `InstructorID`  
6    FOREIGN KEY (`InstructorID`)  
7      REFERENCES `FinalProject`.`Instructors` (`InstructorID`)  
8      ON DELETE NO ACTION  
9      ON UPDATE NO ACTION;  
10
```

Adds the foreign key “InstructorID” to the Courses table

```
1  CREATE TABLE `FinalProject`.`Grades` (  
2    `StudentID` INT NOT NULL,  
3    `CourseID` VARCHAR(45) NOT NULL,  
4    `InstructorID` INT NOT NULL,  
5    `NumGrade` INT NOT NULL);  
6
```

SQL create table command for Grades table

```
1  ALTER TABLE `FinalProject`.`Grades`  
2  ADD CONSTRAINT `StudentID`  
3    FOREIGN KEY (`StudentID`)  
4      REFERENCES `FinalProject`.`Student` (`StudentID`)  
5      ON DELETE NO ACTION  
6      ON UPDATE NO ACTION,  
7  ADD CONSTRAINT `CourseID`  
8    FOREIGN KEY (`CourseID`)  
9      REFERENCES `FinalProject`.`Courses` (`CourseID`)  
10     ON DELETE NO ACTION  
11     ON UPDATE NO ACTION,  
12  ADD CONSTRAINT `InstructorID1`  
13    FOREIGN KEY (`InstructorID`)  
14      REFERENCES `FinalProject`.`Instructors` (`InstructorID`)  
15      ON DELETE NO ACTION  
16      ON UPDATE NO ACTION;  
17
```

Adds the foreign keys “InstructorID”, “CourseID”, and “StudentID” to the Grades table

### Part B: Sample Data (2points)

- Populate your database with sample data. Each relation should contain at least 6 tuples. Make sure that the populated data is suitable for the type of queries in Part C.

Student						
<u>StudentID</u>	Fname	Lname	City	Phone	DateOfBirth	Program
10001	Rayyan	Mohammed	Toronto	416-875-2145	2001-10-23	Engineering
10002	Daniyal	Khan	Toronto	647-854-7865	2001-03-26	Business
10003	Charles	Olagunju	Pickering	209-451-8765	2001-07-24	Health Science
10004	Yousif	Sarmad	Oshawa	905-324-7851	1992-01-01	Engineering
10005	Zaeem	Khalid	Oshawa	905-789-3214	1990-05-12	Business
10006	John	Doe	Mississauga	209-147-8532	1983-12-15	Nursing
Courses						
<u>CourseID</u>	CourseName	CourseDelivery	Capacity	ExamDates	InstructorID	
ENG3200	Intro to Engineering	In Person	140	2021-12-15	20001	
ENG3850	Engineering Economics	In Person	140	2021-12-12	20001	
BUS2840	Accounting Principles	Hybrid	200	2021-12-13	20002	
BUS1470	Intro to Business	Online	500	2021-12-09	20002	
HSCI4800	Life Science Advanced	Hybrid	200	2021-12-12	20004	
NUR1000	Nursing Principles	In Person	100	2021-12-15	20006	
HSCI1000	Intro to Life Science	Online	500	2021-12-05	20003	
NUR4000	Long Term Care	Hybrid	200	2021-12-07	20005	
Instructors						
<u>InstructorID</u>	Fname	Lname				
20001	Khalid	Hafeez				
20002	Usman	Aziz				
20003	Farhan	Mohammed				
20004	David	Suzuki				
20005	LeBron	James				
20006	Franklin	Torgbo				
Grades						
<u>StudentID</u>	<u>CourseID</u>	<u>InstructorID</u>	NumGrade			
10001	ENG3200	20001	83			
10001	ENG3850	20001	74			
10002	BUS2840	20002	91			
10002	BUS1470	20002	89			
10003	HSCI4800	20004	84			
10003	HSCI1000	20006	62			
10004	ENG3200	20001	53			
10004	ENG3850	20001	66			
10005	BUS2840	20002	41			
10005	BUS1470	20002	86			
10006	NUR1000	20006	33			
10006	NUR4000	20005	27			

## Part C: Views

(3points)

- Create (write English description and SQL syntax) of 10 views that a user of the database system would find useful. From these 10 views, the first 5 are common for all groups, and you can create your own views for the remaining ones.
- View 1: Computes a join of at least three tables
- View 2: Uses nested queries with the ANY or ALL operator and uses a GROUP BY clause
- View3: A correlated nested query
- View 4: Uses a FULL JOIN
- View 5: Uses nested queries with any of the set operations UNION, EXCEPT, or INTERSECT

### Custom Views 1:

The screenshot shows a database interface with several code snippets and a result grid.

Code Snippets:

```
1 # CUSTOM VIEW 1
2 • SELECT * FROM FinalProject.Grades
3 WHERE NumGrade > 50
4 ORDER BY StudentID;
5
6 # CUSTOM VIEW 2
7 • SELECT * FROM FinalProject.Student
8 WHERE City IN ("Toronto", "Mississauga")
9 ORDER BY Fname;
10
11 # CUSTOM VIEW 3
12 • SELECT * FROM FinalProject.Courses
13 WHERE CourseDelivery = "In Person"
14 AND Capacity > 100;
15
16 # CUSTOM VIEW 4
17 • SELECT sum(capacity)
18 FROM FinalProject.Courses
19 ◇ 20:4
```

Result Grid:

StudentID	CourseID	InstructorID	NumGrade
10001	ENG3850	20001	74
10001	ENG3200	20001	83
10002	BUS1470	20002	89
10002	BUS2840	20002	91
10003	HSC1000	20006	62
10003	HSC1000	20004	84
10004	ENG3200	20001	53
10004	ENG3850	20001	86
10005	BUS1470	20002	86

Action Output:

Time	Action	Response	Duration / Fetch Time
63 05:55:16	SELECT * FROM FinalProject.Grades WHERE NumGrade > 50 ORDER B...	9 row(s) returned	0.00032 sec / 0.0000...

This view displays the Grades table for the students who have a grade higher than 50. It orders the table by the StudentID. This view is useful to our project because it helps us to see which students are passing the courses.

## Custom Views 2:

```

1  # CUSTOM VIEW 1
2  SELECT * FROM FinalProject.Grades
3  WHERE NumGrade > 50
4  ORDER BY StudentID;
5
6  # CUSTOM VIEW 2
7  SELECT * FROM FinalProject.Student
8  WHERE City IN ("Toronto", "Mississauga")
9  ORDER BY Fname;
10
11 # CUSTOM VIEW 3
12 SELECT * FROM FinalProject.Courses
13 WHERE CourseDelivery = "In Person"
14 AND Capacity > 100;
15
16 # CUSTOM VIEW 4
17 SELECT sum(capacity)
18 FROM FinalProject.Courses
100%  16:59  16:59

```

**Result Grid**

StudentID	Fname	Lname	City	Phone	DateOfBirth	Program
10002	Danyal	Khan	Toronto	647-854-7865	2001-03-26 00:00:00	Business
10006	John	Doe	Mississauga	209-147-8532	1983-12-15 00:00:00	Nursing
10001	Rayan	Mohammed	Toronto	416-875-2145	2001-10-23 00:00:00	Engineering
HULL	HULL	HULL	HULL	HULL	HULL	
HULL	HULL	HULL	HULL	HULL	HULL	
HULL	HULL	HULL	HULL	HULL	HULL	
HULL	HULL	HULL	HULL	HULL	HULL	

Student 24

Action Output

Time	Action	Response	Duration / Fetch Time
06 05:59:20	SELECT * FROM FinalProject.Student WHERE City IN ("Toronto", "Mississauga")	3 row(s) returned	0.00028 sec / 0.0000...

This view gets information from the Student table and if they live in the city Toronto or Mississauga the data is ordered by their first name. This view is key to our project because it allows us to see which students need to commute to classes.

## Custom Views 3:

```

1  # CUSTOM VIEW 1
2  SELECT * FROM FinalProject.Grades
3  WHERE NumGrade > 50
4  ORDER BY StudentID;
5
6  # CUSTOM VIEW 2
7  SELECT * FROM FinalProject.Student
8  WHERE City IN ("Toronto", "Mississauga")
9  ORDER BY Fname;
10
11 # CUSTOM VIEW 3
12 SELECT * FROM FinalProject.Courses
13 WHERE CourseDelivery = "In Person"
14 AND Capacity > 100;
15
16 # CUSTOM VIEW 4
17 SELECT sum(capacity)
18 FROM FinalProject.Courses
100%  20:14  06:02:51

```

**Result Grid**

CourseID	CourseName	CourseDelivery	Capacity	ExamDates	InstructorID
ENG3200	Intro to Engineering	In Person	140	2021-12-15 00:00:00	20001
ENG3850	Engineering Economics	In Person	140	2021-12-12 00:00:00	20001
HULL	HULL	HULL	HULL	HULL	HULL
HULL	HULL	HULL	HULL	HULL	HULL
HULL	HULL	HULL	HULL	HULL	HULL

Courses 25

Action Output

Time	Action	Response	Duration / Fetch Time
06 06:02:51	SELECT * FROM FinalProject.Courses WHERE CourseDelivery = "In Person"	2 row(s) returned	0.00028 sec / 0.0000...

This view gets information from the Courses table and displays the in person classes with a capacity greater than 100. This view is useful because we can see which classes are in person and how many are attending with new covid laws and restrictions.

## Custom Views 4:

The screenshot shows the Oracle SQL Developer interface. The code editor at the top contains the following SQL code:

```
16 # CUSTOM VIEW 4
17 • SELECT sum(capacity)
18 From FinalProject.Courses
19 WHERE CourseDelivery = "Online";
20
21 # CUSTOM VIEW 5
22 • SELECT * FROM FinalProject.Courses
23 WHERE CourseDelivery = "In Person"
24 ORDER BY ExamDates;
25
26 # VIEW 1
27 • SELECT e.studentID, s.NumGrade, d.CourseID
28 FROM (Student e JOIN Grades s ON e.StudentID = s.StudentID)
29 JOIN Courses d ON s.CourseID = d.CourseID;
30
31 # VIEW 2
32 • SELECT StudentID, CourseID, NumGrade
33 FROM Grades
```

The results grid below shows a single row with the value 1000 for the sum of capacity.

sum(c...)
1000

Action Output shows the executed query and its duration:

Time	Action	Response	Duration / Fetch Time
06:08:28	SELECT sum(capacity) From FinalProject.Courses WHERE CourseDeliv...	1 row(s) returned	0.00030 sec / 0.000...

This view checks the total capacity under the Courses table where the classes are online. It's useful to see which classes are online and determine the amount of students attending.

## Custom Views 5:

The screenshot shows the Oracle SQL Developer interface. The code editor at the top contains the following SQL code:

```
20
21 # CUSTOM VIEW 5
22 • SELECT * FROM FinalProject.Courses
23 WHERE CourseDelivery = "In Person"
24 ORDER BY ExamDates;
```

The results grid below displays three rows of course information.

CourseID	CourseName	CourseDelivery	Capacity	ExamDates	InstructorID
ENG3850	Engineering Economics	In Person	140	2021-12-12 00:00:00	20001
ENG3200	Intro to Engineering	In Person	140	2021-12-15 00:00:00	20001
NUR1000	Nursing Principles	In Person	100	2021-12-15 00:00:00	20006
NULL	NULL	NULL	NULL	NULL	NULL

Action Output shows the executed query and its duration:

Time	Action	Response	Duration / Fetch Time
06:12:30	SELECT * FROM FinalProject.Courses WHERE CourseDelivery = "In Per..."	3 row(s) returned	0.00025 sec / 0.0000...

This view gets information from the Courses table and displays the in person classes, ordering them by the exam dates. This view is useful for students that need to go in on the dates published.

### View 1:

```
26 # VIEW 1
27 • SELECT e.studentID, s.NumGrade, d.CourseID
28 FROM (Student e JOIN Grades s ON e.StudentID = s.StudentID)
29   JOIN Courses d ON s.CourseID = d.CourseID;
30
```

Result Grid | Filter Rows: Search Export: | Result Grid  
studentID NumGrade CourseID |  
10001 74 ENG3850  
10001 83 ENG3200  
10002 89 BUS1470  
10002 91 BUS2840  
10003 62 HSCI1000  
10003 84 HSCI4800  
10004 53 ENG3200  
10004 66 ENG3850  
10005 41 BUS2840  
10005 86 BUS1470  
10006 27 NUR4000  
10006 33 NUR1000

Action Output | Time Action Response Duration / Fetch Time  
70 06:14:28 SELECT e.studentID, s.NumGrade, d.CourseID FROM (Student e JOIN... 12 row(s) returned 0.00037 sec / 0.0000...

This view joins Students, Grades and Courses table together while displaying StudentID, NumGrade and CourseID Points.

### View 2:

```
31 # VIEW 2
32 • SELECT StudentID, CourseID, NumGrade
33   FROM Grades
34   WHERE NumGrade > ANY ( SELECT NumGrade
35                         FROM Grades
36                         WHERE StudentID = 10001 )
37 GROUP BY NumGrade;
38
```

Result Grid | Filter Rows: Search Edit: Export/Import: | Result Grid  
StudentID CourseID NumGrade |  
10001 ENG3200 83  
10003 HSCI4800 84  
10005 BUS1470 86  
10002 BUS1470 89  
10002 BUS2840 91

Action Output | Time Action Response Duration / Fetch Time  
71 06:18:03 SELECT StudentID, CourseID, NumGrade FROM Grades WHERE NumG... 5 row(s) returned 0.00029 sec / 0.0000...

This view displays any student who has a higher grade than the lowest grade for student “10001” which is 74.

### View 3:

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
39 # VIEW 3
40 • SELECT Capacity, CourseName, InstructorID
41   FROM Courses
42   WHERE Capacity > (SELECT sum(Capacity)
43     FROM Courses
44     WHERE InstructorID = 20001);
45
```

The results grid displays two rows of data:

Capacity	CourseName	InstructorID
500	Intro to Business	20002
500	Intro to Life Science	20003

Action Output shows the query and its execution details:

Time	Action	Response	Duration / Fetch Time
06:19:34	SELECT Capacity, CourseName, InstructorID FROM Courses WHERE C...	2 row(s) returned	0.00036 sec / 0.000...

Displays the Capacity, InstructorID and CourseName for the instructors who have a higher total capacity than instructor “20001” which is 380.

### View 4:

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
46 # VIEW 4
47 • SELECT t1.CourseDelivery, t2.Lname
48   FROM Courses AS t1 LEFT JOIN Instructors AS t2
49   ON t1.InstructorID = t2.InstructorID
50 UNION
51   SELECT t1.CourseDelivery, t2.Lname
52   FROM Courses AS t1 RIGHT JOIN Instructors AS t2
53   ON t2.InstructorID = t1.InstructorID
54 ORDER BY Lname;
55
```

The results grid displays seven rows of data:

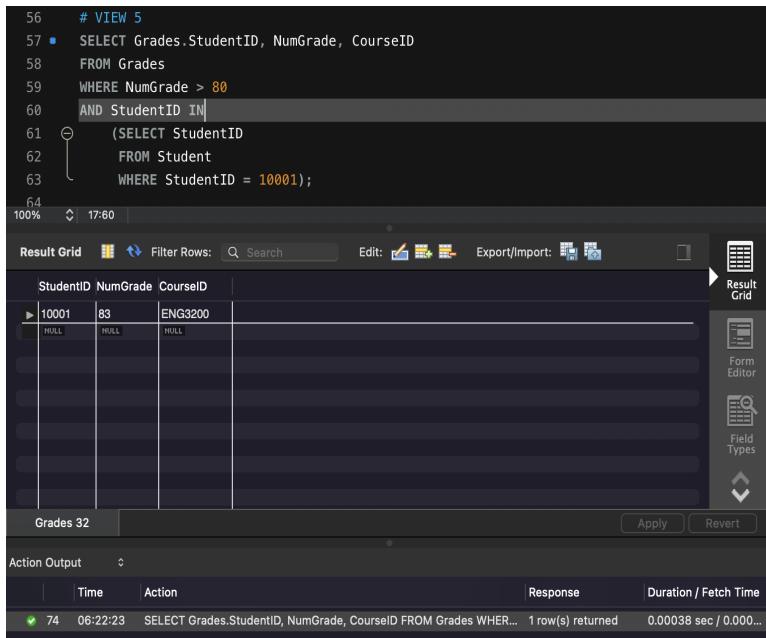
CourseDelivery	Lname
Online	Aziz
Hybrid	Aziz
In Person	Hafeez
Hybrid	James
Online	Mohammed
Hybrid	Suzuki
In Person	Torgbo

Action Output shows the query and its execution details:

Time	Action	Response	Duration / Fetch Time
06:20:32	SELECT t1.CourseDelivery, t2.Lname FROM Courses AS t1 LEFT JOIN I...	7 row(s) returned	0.00044 sec / 0.000...

Displays Left, Right Join and Union to emulate a Full Join in MySQL Workbench. Displays course delivery and last name by combining Courses table and Instructors table.

### View 5:



The screenshot shows the MySQL Workbench interface. The SQL editor window contains the following code:

```
56 # VIEW 5
57 • SELECT Grades.StudentID, NumGrade, CourseID
58 FROM Grades
59 WHERE NumGrade > 80
60 AND StudentID IN
61 (SELECT StudentID
62 FROM Student
63 WHERE StudentID = 10001);
64
```

The result grid shows one row of data:

StudentID	NumGrade	CourseID
10001	83	ENG3200

The status bar at the bottom indicates "Action Output" with the following details:

Time	Action	Response	Duration / Fetch Time
06:22:23	SELECT Grades.StudentID, NumGrade, CourseID FROM Grades WHERE...	1 row(s) returned	0.00038 sec / 0.000...

Displays emulation of an Intercept in MySQL Workbench by using IN. Displays StudentID,NumGrade and CourseID where the student with id “10001” got a higher grade than 80.

### **ALL VIEWS:**

```
1 # CUSTOM VIEW 1
2 • SELECT * FROM FinalProject.Grades
3 WHERE NumGrade > 50
4 ORDER BY StudentID;
5
6 # CUSTOM VIEW 2
7 • SELECT * FROM FinalProject.Student
8 WHERE City IN ("Toronto", "Mississauga")
9 ORDER BY Fname;
10
11 # CUSTOM VIEW 3
12 • SELECT * FROM FinalProject.Courses
13 WHERE CourseDelivery = "In Person"
14 AND Capacity > 100;
15
16 # CUSTOM VIEW 4
17 • SELECT sum(capacity)
18 From FinalProject.Courses
19 WHERE CourseDelivery = "Online";
20
21 # CUSTOM VIEW 5
22 • SELECT * FROM FinalProject.Courses
23 WHERE CourseDelivery = "In Person"
24 ORDER BY ExamDates;
```

```
26      # VIEW 1
27 •  SELECT e.studentID, s.NumGrade, d.CourseID
28   FROM (Student e JOIN Grades s ON e.StudentID = s.StudentID)
29     JOIN Courses d ON s.CourseID = d.CourseID;
30
31      # VIEW 2
32 •  SELECT StudentID, CourseID, NumGrade
33   FROM Grades
34   WHERE NumGrade > ANY ( SELECT NumGrade
35     FROM Grades
36     WHERE StudentID = 10001)
37   GROUP BY NumGrade;
38
39      # VIEW 3
40 •  SELECT Capacity, CourseName, InstructorID
41   FROM Courses
42   WHERE Capacity > (SELECT sum(Capacity)
43     FROM Courses
44     WHERE InstructorID = 20001);
45
46      # VIEW 4
47 •  SELECT t1.CourseDelivery, t2.Lname
48   FROM Courses AS t1 LEFT JOIN Instructors AS t2
49   ON t1.InstructorID = t2.InstructorID
50   UNION
51   SELECT t1.CourseDelivery, t2.Lname
52   FROM Courses AS t1 RIGHT JOIN Instructors AS t2
53   ON t2.InstructorID = t1.InstructorID
54   ORDER BY Lname;
```

```
56      # VIEW 5
57 •  SELECT Grades.StudentID, NumGrade, CourseID
58   FROM Grades
59   WHERE NumGrade > 80
60   AND StudentID IN
61   (SELECT StudentID
62     FROM Student
63     WHERE StudentID = 10001);
64
```

Part D: E-R diagram: (5points)

- Create an ER schema diagram for your project database.

