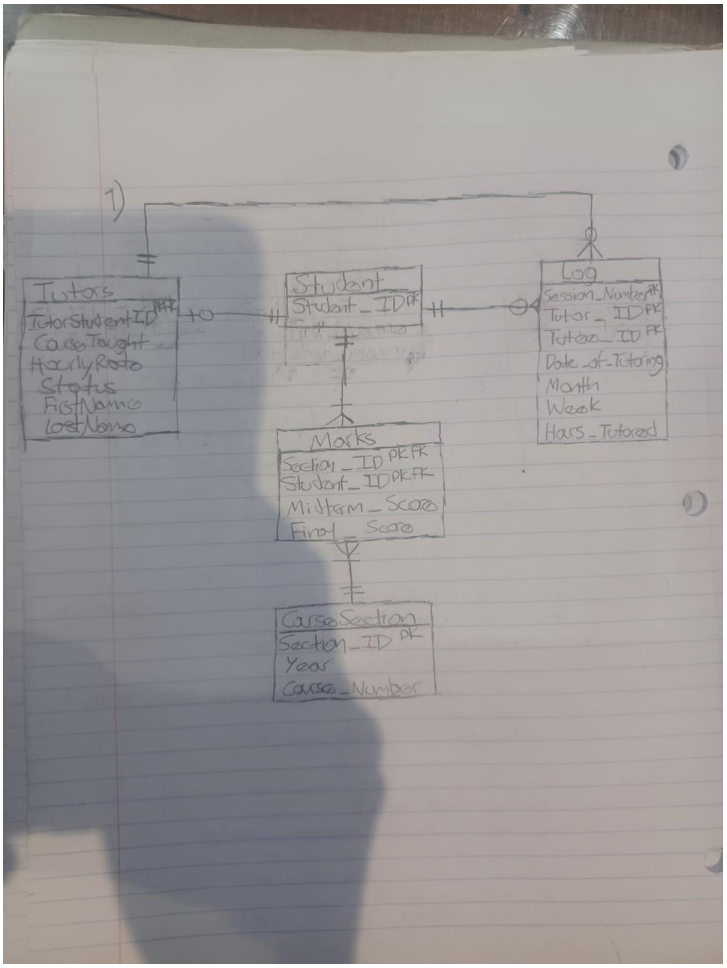
Ashwin Narayan and Joshua Jowers

1. -AN



Referential Integrity Constraints:

Student_ID in Tutors must exist in Student_ID in Student and must not be null

Tutor_ID in Log must exist in TutorStudentID in Tutors and must not be null

Tutee_ID in Log must exist in Student_ID in Student and must not be null

Section_ID in Marks must exist in Section_ID in CourseSection and must not be null

Student_ID in Marks must exist in Student_ID in Student and must not be null

Assumptions:

- Every tutor is a student with a student ID
- Each tutor id is a student id belonging to the student who is tutoring

- Repeated student ids in the Tutor table are assumed to be data errors

Our model satisfies 2NF but not 3NF, because we have a functional dependency in our Log table. We chose to denormalize our database so that we did not have to create separate tables for Date, Month, and Week. Had we chosen to design our model in 3NF, we would have had to split up Date and Month and Date and Week. This would have made the design unnecessarily complicated for users who probably prefer seeing the date, month, and week neatly organized in the same table.

b) -JJ

Narrative Description:

There are many students enrolled at a university. Each student is identified by a unique student ID number. Students can take several courses at a time. At the end of each course, the midterm and final grade for each student is recorded with the section ID of the course and the student ID of the student who the marks belong to. Each course can be offered multiple times across different years, and each time the course is offered, a course section ID is given. This is what is used in recording the students' grades. Some of these students serve as tutors for other students in classes that they have passed. The tutors are identified by their student ID. During each tutoring session, one tutor tutors one student. Each of these tutoring sessions is recorded as an entry in a log. Each entry in the log is identified by a unique session number. The log also keeps track of the student IDs of both the tutor and tutee as well as the date that the sessions took place and the length of the session in hours.

2. -JJ

a. In creation of the database, we were unable to use one database create statement to create the database, so we used a few steps to implement our design. We started by using the data Grip functionality in Intellij to import the original 3 csv files. Then we used the below SQL statement to create the student table.

```
Create Table STUDENT AS
select distinct "Student_ID Number"
from mark
order by "Student_ID Number";
```

Then we used this next SQL statement to create the course_section table, but this was without the surrogate key. We then went and implemented the surrogate key into this table using Excel then reuploaded this table again using DataGrip

```
create table Course_Sections as
select course_section."Section_ID", "Student_ID Number", "Exam1_Score", "Exam2_Score"
from course_section, marks
where Course_Section."Course_Number" = marks."Course_Number"
and Course_Section."Year" = marks."Year";
```

The next modification we made was to remove the year and course number from the marks table by creating a new table called mark that contained the course_section in place of course number and year. This was accomplished with this statement:

```
create table mark as
    select course_section.section_number, marks."Student_ID Number", "Midterm_Score",
"Final_Score"
    from course_section, marks
    where course_section."Course_Number" = marks."Course_Number"
and course_section."Year" = marks."Year";
```

Then, we dropped the marks table from the database, then implemented the referential integrity constraints above and data types using the user interface in IntelliJ. For data types, we used numeric values for midterm and final score, hourly rate, and length of session. The rest of the columns had a data time of text or char.

Query 1- AN

```
select "First_Name" as First_Name, "Last_Name" as Last_Name
from tutors
where "Course_Taught" = 'CS1TA3'
order by "Last_Name"
limit 5;
```

first_name	last_name
Stephen	Alexander
Dennis	Anderson
Francesca	Bomberry
Shirley	Boyle
Alice	Cardwell

Query 2- AN

```
select "First_Name" as First_Name, "Last_Name" as Last_Name, "Hourly_Rate" as Hourly_Wage
from tutors
where "Course_Taught" = 'CS1TA3'
and "Hourly_Rate" = 17.50
order by "Last_Name"
limit 5;
```

first_name	last_name	hourly_wage
Stephen	Alexander	17.5
Francesca	Bomberry	17.5
Alice	Cardwell	17.5
Debra	Hall	17.5
Julie	Kober	17.5

Query 3- AN

```
select "First_Name" as First_Name, "Last_Name" as Last_Name, "Hourly_Rate" as Hourly_Wage
from tutors
where "Course_Taught" = 'CS1TA3'
 and ("Hourly_Rate" = 17.50 or "Hourly_Rate" = 15.00)
```

```
order by "Last_Name"
limit 5;
```

first_name	last_name	hourly_wage
Stephen	Alexander	17.5
Dennis	Anderson	15
Francesca	Bomberry	17.5
Alice	Cardwell	17.5
George	Cooper	15

Query 4-AN

```
select distinct on ("Student_ID Number") "Student_ID Number", "Final_Score"
from mark
where ("Final_Score" > 95 or "Final_Score" <= 60)
 and section_number in
 (
 select course_section.section_number
 from course_section
 where "Year" = '2006')
order by "Student_ID Number"
limit 5;
```

Student_ID Number	Final_Score
60011	58
60012	59
60014	60
60015	52
60021	59

Query 5-AN

```
select distinct "Student_ID Number"
from mark
```

```
where section_number in (select course_section.section_number
 from course_section
 where "Course_Number" in ('CS1TA3', 'CS1FC3', 'CS1MD3')
 and "Year" = '2006')
order by "Student_ID Number"
limit 5;
```

Student_ID Number
60006
60010
60015
60016
60017

Query 6- AN

```
select distinct "Student_ID Number"
from mark
where section_number in (
select course_section.section_number
from course_section
where "Year" = '2006'
and "Course_Number" like 'ENG%')
order by "Student_ID Number"
limit 5
```

Student_ID Number
60004
60006
60009
60010
60011

Query 7- JJ

```
with full_mark as (select *
 from course_section
 join mark on course_section.section_number = mark.section_number)
```

```
select "Student_ID Number" as tutor_id, "Course_Number", "Final_Score"
from full_mark, tutors
where tutors."Course_Taught" = full_mark."Course_Number"
and tutors."Student_ID" = full_mark."Student_ID Number"
and "Final_Score" > 95
order by "Student_ID Number"
limit 5;
```

tutor_id	Course_Number	Final_Score
40007	CS1TA3	96
40032	CS1SA3	98
40053	CS1BA3	98
40143	CS1TA3	98
40202	CS1TA3	98

Query 8- JJ

```
select "Student_ID Number", "Final_Score"
from mark
where mark.section_number in (
   select course_section.section_number
   from course_section
   where "Course_Number" = 'CS1TA3'
    and "Year" = '2006')
order by "Final_Score" desc
limit 5
```

Student_ID Number	Final_Score
60036	98
60081	98
60870	98
60809	98

60709 98

Query 9- AN

```
SELECT AVG("Final_Score")
FROM mark
INNER JOIN course_section cs on mark.section_number = cs.section_number
Where cs."Course_Number" = 'CS1TA3' and "Year" = 2006;
```

avg
77.2779552715654952

Query 10- AN

```
SELECT AVG("Final_Score"), "Course_Number"
From mark, course_section
Where "Course_Number" LIKE 'CS%'
group by "Course_Number"
limit 5;
```

avg	Course_Number
76.8226168056676531	CS1MD3
76.8226168056676531	CS1SA3
76.8226168056676531	CS1TA3
76.8226168056676531	CS1MA3
76.8226168056676531	CS1FC3

Query 12- AN

```
with cte as (select *
 from course_section
```

```
join mark on course_section.section_number = mark.section_number
where course_section."Year" = '2006'
and course_section."Course_Number" like 'CS%')
select distinct "Student_ID Number"
from cte
where "Final_Score" < 65
and "Student_ID Number" not in (select "Tutee_Student_ID"
from log)
limit 5;
```

Student_ID Number
60021
60040
60047
60057
60083

Query 13- AN

```
select sum("Hourly_Rate" * "Length_Of_Session") as total_money_made
 from tutors, log
 where "Tutor_Student_ID" = tutors."Student_ID"
 and "First_Name" = 'Carolyn'
 and "Last_Name" = 'Mitchell'
```

total_money_made
1155

Query 14- AN

```
with total_hours as (select sum("Length_Of_Session") as total_tutoring_hours
 from log),
 november_hours as (
 select sum("Length_Of_Session") as november_tutoring_hours
 from log
 where "Month" = 'NOV'
 )
 select *
 from total_hours, november_hours
```

total_tutoring_hours	november_tutoring_hours
5530.5	1007

Query 15- AN

```
with total_hours as (
    select sum("Length_Of_Session") as total_tutoring_hours
    from log
),
     november_hours as (
         select sum("Length_Of_Session") as november_tutoring_hours
         from log
         where "Month" = 'NOV'
     )
   select (CAST(november_tutoring_hours AS numeric) / total_tutoring_hours) AS
november_failure_ratio
from total_hours, november_hours;
```

november_hours_ratio
0.18208118614953440014

Query 16- JJ

```
SELECT mark."Student_ID Number","Final_Score", avg_table.average
 FROM mark join course_section on mark.section_number = course_section.section_number join
 (
 SELECT AVG("Final_Score") AS average
 FROM mark join course_section on course_section.section_number = mark.section_number
 where course_Section."Course_Number" = 'CS1TA3'
 and Course_Section."Year" = '2006'
 ) AS avg_table
 on True
 where course_Section."Course_Number" = 'CS1TA3'
 and Course_Section."Year" = '2006'
 order by "Final_Score"
 limit 5;
```

Student_ID Number	Final_Score	average
60546	52	77.2779552715654952

60015	52	77.2779552715654952
60526	53	77.2779552715654952
60468	55	77.2779552715654952
60885	56	77.2779552715654952

Query 17-JJ

```
SELECT mark."Student_ID Number","Final_Score", avg_table.average
 FROM mark join course_section on mark.section_number = course_section.section_number join
 (
 SELECT AVG("Final_Score") AS average
 FROM mark join course_section on course_section.section_number = mark.section_number
 where course_Section."Course_Number" = 'CS1TA3'
 and Course_Section."Year" = '2006'
 ) AS avg_table
 on True
where course_Section."Course_Number" = 'CS1TA3'
 and Course_Section."Year" = '2006'
 and avg_table.average > "Final_Score"
 order by "Final_Score"
limit 5;
```

Student_ID Number	Final_Score	average
60015	52	77.2779552715654952
60546	52	77.2779552715654952
60526	53	77.2779552715654952
60468	55	77.2779552715654952
60851	56	77.2779552715654952

Query 18-JJ

```
select "Student_ID Number", "Final_Score"
 from mark
 where mark.section_number in (
 select course_section.section_number
 from course_section
 where "Course_Number" = 'CS1TA3'
 and "Year" = '2006')
 and "Final_Score"
 in (select max("Final_Score")
```

```
from mark
where mark.section_number in
(select course_section.section_number
from course_section
where "Course_Number" = 'CS1TA3'
and "Year" = '2006'))
order by "Final_Score"
limit 5
```

Student_ID Number	Final_Score
60081	98
60870	98
60809	98
60036	98
60709	98

Query 19-JJ- Note: This is assuming that the duplicate student IDs in the tutor table with different names associated with them belong to different tutors and that there is just an issue with the id column. This also assumes that there are no tutors tutoring multiple courses which is backed up by the data.

```
select count("Student_ID")
 from tutors
```

count
128

Query 20- JJ

```
with full_mark as (select *
 from course_section
 join mark on course_section.section_number = mark.section_number)
 select "Student_ID Number"
 from full_mark, tutors
 where tutors."Course_Taught" = full_mark."Course_Number"
 and tutors."Student_ID" = full_mark."Student_ID Number"
 and "Final_Score" <= 90
limit 5
```

Student_ID Number

40749

40734

40617

40381

40839

Query 21- JJ

```
update tutors
 set "Status" = 'Inactive'
 where tutors."Student_ID" in(
 with full_mark as (select *
 from course_section
 join mark on course_section.section_number = mark.section_number)
 select "Student_ID Number"
 from full_mark, tutors
 where tutors."Course_Taught" = full_mark."Course_Number"
 and tutors."Student_ID" = full_mark."Student_ID Number"
 and "Final_Score" <= 90)
```

Query 22- AN

```
select count(distinct "Course_Number")
from course_section
where "Course_Number" like 'CS%'
```

count

6

Query 23- AN

```
select count(distinct "Student_ID Number") cs_tutoring_market
 from mark
where mark.section_number in (
 select course_section.section_number
 from course_section
 where "Course_Number" like 'CS%'
```

```
and "Year" = '2006')
and "Final_Score" < 65;
```

cs_tutoring_market

252

Query 24- AN

```
with cte as (select distinct "Tutor_Student_ID", "Hourly_Rate", "Length_Of_Session",
"Tutee_Student_ID","Month", "Hourly_Rate" * "Length_Of_Session" as total_cost
from tutors,
log
where log."Tutor_Student_ID" = tutors."Student_ID")
select round(sum(total_cost) * 0.15) as January_profit
from cte
where "Month" = 'JAN';
```

january_profit

2103

Query 25- AN

```
with cte as (select distinct "Tutor_Student_ID", "Hourly_Rate", "Length_Of_Session",
"Tutee_Student_ID","Month", "Hourly_Rate" * "Length_Of_Session" as total_cost
from tutors,
log
where log."Tutor_Student_ID" = tutors."Student_ID")
select "Month", round(sum(total_cost) * 0.15) as monthly_profit
from cte
group by "Month"
order by monthly_profit desc
limit 1;
```

Month	monthly_profit
NOV	2358

Query 26 - AN

```
with cte as (select distinct "Tutor_Student_ID", "Hourly_Rate", "Length_Of_Session",
"Tutee_Student_ID","Month", "Hourly_Rate" * "Length_Of_Session" as total_cost
 from tutors,
 log
 where log."Tutor_Student_ID" = tutors."Student_ID")
 select round(sum(total_cost) * 0.15) as total_profit
 from cte
```

total_profit

13077

Query 27 - AN

```
with cte1 as (with cte as (select distinct "Tutor_Student_ID",
 "Hourly_Rate",
 "Length_Of_Session",
 "Tutee_Student_ID",
 "Month",
 "Hourly_Rate" * "Length_Of_Session" as total_cost
 from tutors,
 log
 where log."Tutor_Student_ID" = tutors."Student_ID")
 select round(sum(total_cost) * 0.15) as total_profit
 from cte),
 cte2 as (select distinct count("Student_ID Number") as cs_tutoring_market
 from mark
 where mark.section_number in (
 select course_section.section_number
 from course_section
 where "Course_Number" like 'CS%'
 and "Year" = '2006')
 and "Final_Score" < 65)
 select round(cte1.total_profit / cte2.cs_tutoring_market, 1) as profit_to_market_ratio
 from cte1, cte2;
```

profit_to_market_ratio

45.6

Query 28 - AN

```
with full_mark as (select course_section."Year", Course_Section.section_number,
course_section."Course_Number", "Final_Score"
 from course_section
 join mark on course_section.section_number = mark.section_number
 where "Year" = '2006'
 and "Final_Score" < 65
```

```
 and "Course_Number" like 'ENG%')
 select "Course_Number", count("Final_Score") as tutoring_market
 from full_mark
group by "Course_Number"
 order by tutoring_market desc
 limit 1;
```

Course_Number	tutoring_market
ENG1B03	57

Query 29 - AN

```
with full_mark as (select course_section."Year", Course_Section.section_number,
course_section."Course_Number", "Final_Score"
 from course_section
 join mark on course_section.section_number = mark.section_number
 where "Year" = '2006'
 and "Course_Number" like 'ENG%')
 select "Course_Number", avg("Final_Score") as average_grade
 from full_mark
group by "Course_Number"
 order by average_grade
 limit 1;
```

Course_Number	average_grade
ENG1G03	75.6085526315789474

Query 30 - JJ

```
WITH full_mark AS (
 SELECT
 course_section."Course_Number",
 COUNT(*) AS total
 FROM
 course_section
 JOIN
 mark ON course_section.section_number = mark.section_number
 WHERE
 "Year" = '2006'
 AND "Course_Number" LIKE 'ENG%'
 GROUP BY
 "Course_Number"
 ),
 mark_only_failures AS (
 SELECT
```

```
 course_section."Course_Number",
 COUNT("Final_Score") AS failure_total
 FROM
 course_section
 JOIN
 mark ON course_section.section_number = mark.section_number
 WHERE
 "Year" = '2006'
 AND "Course_Number" LIKE 'ENG%'
 AND "Final_Score" < 65
 GROUP BY
 "Course_Number"
 )

 SELECT
full_mark."Course_Number",
 (CAST(failure_total AS numeric) / total) AS failure_rate
 FROM
mark_only_failures
JOIN
 full_mark ON mark_only_failures."Course_Number" = full_mark."Course_Number"
 ORDER BY
failure_rate DESC
 limit 1
```

Course_Number	failure_rate
ENG1H03	0.19503546099290780142

Query 31 - JJ

```
select count(distinct "Student_ID Number") eng_tutoring_market
 from mark
where mark.section_number in (
 select course_section.section_number
 from course_section
 where "Year" = '2006'
 and "Final_Score" < 65
 and "Course_Number" in ('ENG1H03', 'ENG1G03, ENG1B03'))
```

eng_tutoring_market
55

Query 32- JJ

```
with cte as (
 select count(distinct "Student_ID Number") as eng_tutoring_market
 from mark
 where mark.section_number in (
 select course_section.section_number
 from course_section
 where "Year" = '2006'
 and "Final_Score" < 65
 and "Course_Number" in ('ENG1H03', 'ENG1G03, ENG1B03')))
 select eng_tutoring_market * 45.6 as total_engineering_market_opporunity
 from cte
```

total_engineering_market_opporunity
2508

Query 33 - JJ

```
with full_mark as (select *
 from course_section
 join mark on course_section.section_number = mark.section_number)
 select distinct "Student_ID Number"
 from full_mark
 where "Course_Number" in ('ENG1H03', 'ENG1G03, ENG1B03')
 and "Final_Score" >= 95
 and "Student_ID Number" not in (
 select "Student_ID"
 from tutors
 )limit 5;
```

Student_ID Number
40015
40261
40272
40276
40280

Query 34- JJ

```
select "Tutor_Student_ID", sum("Length_Of_Session") as hours_worked
from log
where "Month" = 'MAR'
and "Week Number" > 18
group by "Tutor_Student_ID"
order by hours_worked desc
limit 1
```

Tutor_Student_ID	hours_worked
50490	21