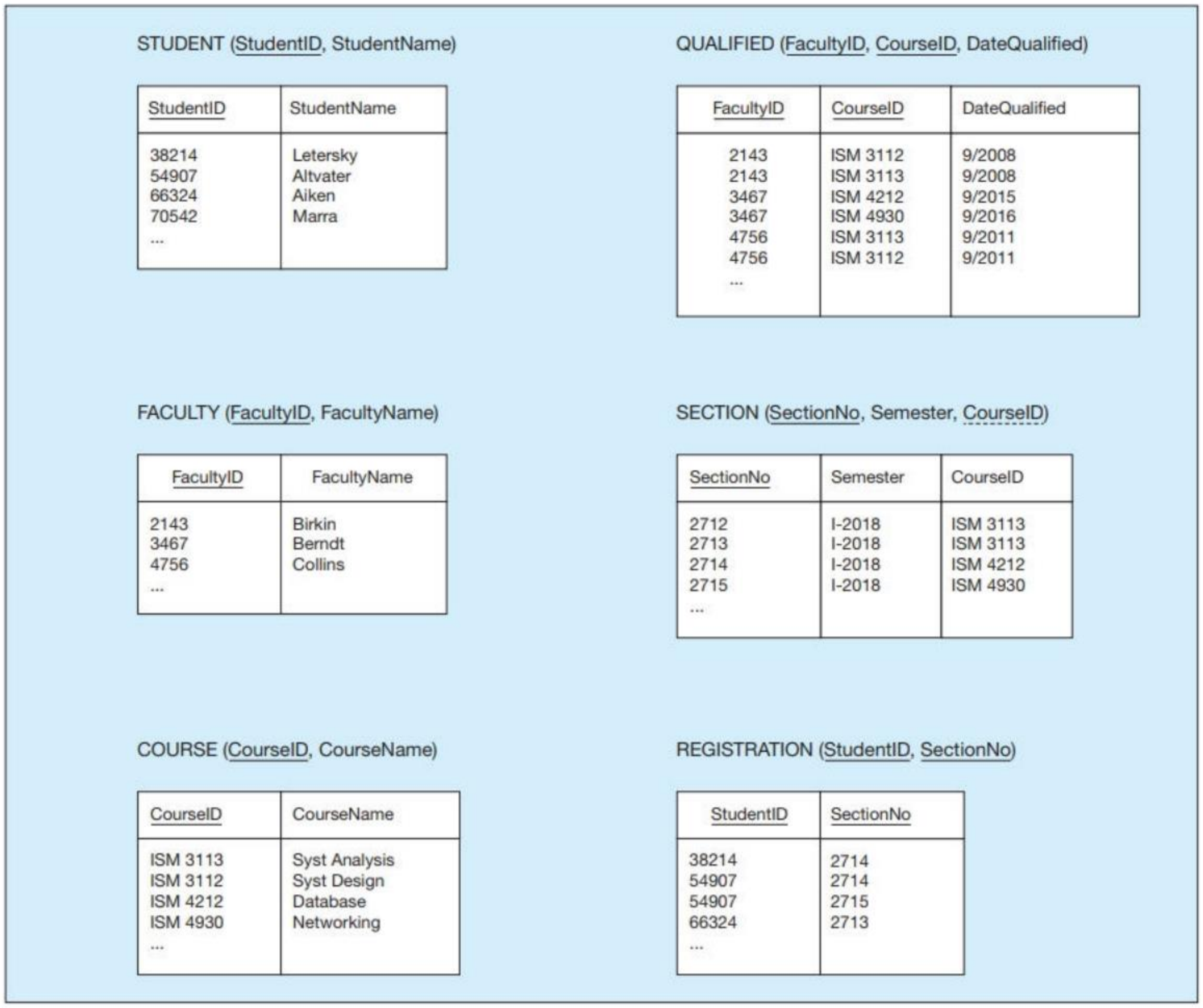
Homework 6

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

Q1. Use SQL to answer the following questions by referring to the DB shown below



STUDENT (StudentID, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
...	

QUALIFIED (FacultyID, CourseID, DateQualified)

FacultyID	CourseID	DateQualified
2143	ISM 3112	9/2008
2143	ISM 3113	9/2008
3467	ISM 4212	9/2015
3467	ISM 4930	9/2016
4756	ISM 3113	9/2011
4756	ISM 3112	9/2011
...		

FACULTY (FacultyID, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
...	

SECTION (SectionNo, Semester, CourseID)

SectionNo	Semester	CourseID
2712	I-2018	ISM 3113
2713	I-2018	ISM 3113
2714	I-2018	ISM 4212
2715	I-2018	ISM 4930
...		

COURSE (CourseID, CourseName)

CourseID	CourseName
ISM 3113	Syst Analysis
ISM 3112	Syst Design
ISM 4212	Database
ISM 4930	Networking
...	

REGISTRATION (StudentID, SectionNo)

StudentID	SectionNo
38214	2714
54907	2714
54907	2715
66324	2713
...	

1. write SQL to answer the following questions [15]

a. Display the course ID and course name for all courses with an ISM prefix.

b. Display the numbers and names of all courses for which Professor Berndt has been qualified.

c. Display the class roster, including student name, for all students enrolled in section 2714 of ISM 4212.

ANSWER:

a.

```
SELECT CourseID, CourseName
FROM COURSE
WHERE CourseID LIKE  ‘%ISM’;
```

b.

```
SELECT COURSE. CourseID, COURSE. CourseName
FROM COURSE, FACULTY, QUALIFIED
WHERE COURSE. CourseID = QUALIFIED. CourseID
AND FACULTY.FacultyID = QUALIFIED. FacultyID
AND FACULTY.FacultyName = ‘Berndt’;
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT c.CourseID, c.CourseName
FROM COURSE AS c, FACULTY AS f, QUALIFIED AS q
WHERE c. CourseID = q. CourseID
AND f.FacultyID = q. FacultyID
AND f.FacultyName = ‘Berndt’;
```

(Or)
(We can use join statement):

```
SELECT c.CourseID, c.CourseName
FROM Qualified q
INNER JOIN COURSE c ON c.CourseID = q.CourseId
INNER JOIN FACULTY f ON f.FacultyID = q.FacultyID
WHERE f.FacultyName  = ‘Berndt’;
```

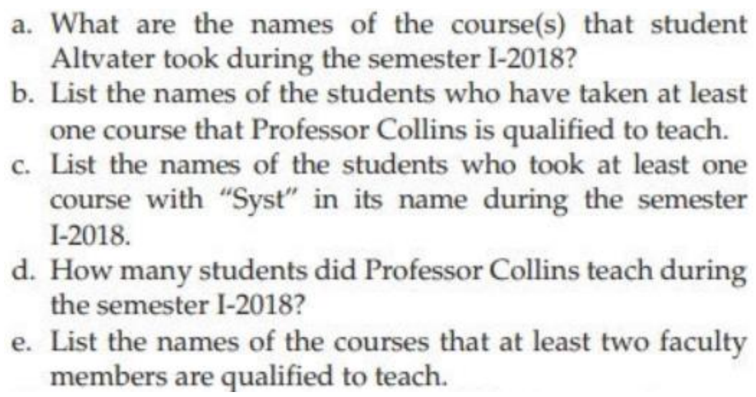
c.
(Using ‘AS’ to shorten the query & improve readability):

```
SELECT s.StudentID, s.StudentName, sn.CourseID, r.SectionNo
FROM STUDENT AS s, SECTION AS sn, REGISTRATION AS r
WHERE s.StudentID = r.StudentID
AND sn.SectionNo = r.SectionNo
AND sn.CourseID = ‘ISM 4214’
AND r.Section No = ‘2714’;
```

(Or)
(We can use join statement):

```
SELECT s.StudentID, s.StudentName, sn.CourseID, r.SectionNo
FROM REGISTRATION r
INNER JOIN STUDENT s ON s.StudentID = r.StudentID
INNER JOIN SECTION sn ON sn.SectionNo = r.SectionNo
WHERE r.SectionNo = 2714;
```

2. write SQL statements [25]



a. What are the names of the course(s) that student Altvater took during the semester I-2018?
b. List the names of the students who have taken at least one course that Professor Collins is qualified to teach.
c. List the names of the students who took at least one course with "Syst" in its name during the semester I-2018.
d. How many students did Professor Collins teach during the semester I-2018?
e. List the names of the courses that at least two faculty members are qualified to teach.

ANSWER:

a.

```
SELECT COURSE.CourseName
FROM COURSE, STUDENT, SECTION, REGISTRATION
WHERE STUDENT.StudentID = REGISTRATION. StudentID
AND COURSE.CourseID = SECTION.CourseID
AND SECTION.SectionNo = REGISTRATION. SectionNo
AND STUDENT.StudentName = ‘Altvater’
AND SECTION.Semester = ‘I-2018’;
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT c.CourseName
FROM COURSE AS c, STUDENT AS s, SECTION AS sn, REGISTRATION AS r
WHERE s.StudentID = r. StudentID
AND c.CourseID =sn.CourseID
AND sn.SectionNo = r. SectionNo
AND s.StudentName = ‘Altvater’
AND sn.Semester = ‘I-2018’;
```

(Or)
(We can use join statement):

```
SELECT c.CourseName
FROM STUDENT s
NATURAL JOIN REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN COURSE c
WHERE s.StudentName = 'Altvater' AND sn.Semester = 'I-2018';
```

(Note: We can use natural join here as it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

b. (as it is asked ‘at least one course’, we need to use distinct as ‘distinct’ eliminates duplicate values)

```
SELECT  DISTINCT  STUDENT.StudentName
FROM  STUDENT, FACULTY, REGISTRATION, SECTION, QUALIFIED
WHERE STUDENT.StudentID =  REGISTRATION. StudentID
AND  FACULTY.FacultyID = QUALIFIED. FacultyID
AND  REGISTRATION.SectionNo = SECTION. SectionNo
AND  SECTION.CourseID =  QUALIFIED. CourseID
AND  FACULTY.FacultyName = ‘Collins’;
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT  DISTINCT  s.StudentName
FROM  STUDENT AS s, FACULTY AS f, REGISTRATION AS r, SECTION AS
sn, QUALIFIED AS q
WHERE  s.StudentID =  r. StudentID
AND  f.FacultyID = q. FacultyID
AND  r.SectionNo = sn. SectionNo
AND  sn.CourseID =  q. CourseID
AND  f.FacultyName = ‘Collins’;
```

(Or)
(We can use join statement):

```
SELECT DISTINCT s.StudentName
FROM STUDENT s
NATURAL JOIN REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN COURSE c
NATURAL JOIN FACULTY f
NATURAL JOIN QUALIFIED q
WHERE f.FacultyName = 'Collins';
```

(Note: We can use natural join here as the it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

c. (as it is asked ‘at least one course’, we need to use distinct as distinct eliminates duplicate values)

```
SELECT  DISTINCT  STUDENT.StudentName
FROM STUDENT, COURSE, SECTION, REGISTRATION
WHERE  STUDENT.StudentID = REGISTRATION. StudentID
AND  COURSE.CourseID = SECTION. CourseID
AND  REGISTRATION.SectionNo = SECTION. SectionNo
AND   COURSE.CourseName LIKE   ‘%Syst%’                (# syst can be at start/end)
AND   SECTION.Semester = ‘I-2018’;
```

(Or)

(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT  DISTINCT  s.StudentName
FROM STUDENT AS s, COURSE AS c, SECTION AS sn, REGISTRATION AS r
WHERE  s.StudentID = r. StudentID
AND  c.CourseID = sn. CourseID
AND  r.SectionNo = sn. SectionNo
AND   c.CourseName LIKE   ‘%Syst%’                (# syst can be at start/end)
AND   sn.Semester = ‘I-2018’;
```

(Or)

(We can use join statement):

```
SELECT DISTINCT s.StudentName
FROM STUDENT s
NATURAL JOIN REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN COURSE c
WHERE c.CourseName LIKE '%Syst%' AND sn.Semester='I-2018';
```

(Note: We can use natural join here as the it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

d. (As it is asked ‘how many students’, we need to use count(student_id)):

```
SELECT  COUNT (REGISTRATION_StudentID) AS  Num_Of_Students
FROM     REGISTRATION, SECTION, QUALIFIED, FACULTY
WHERE  FACULTY.FacultyID =  QUALIFIED. FacultyID
AND    REGISTRATION.SectionNo =  SECTION. SectionNo
AND    QUALIFIED.CourseID =  SECTION. CourseID
AND    FACULTY.FacultyName = ‘Collins’
AND    SECTION.Semester = ‘I-2018’;
```

(Or)

(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT  COUNT (r.StudentID) AS  Num_Of_Students
FROM    REGISTRATION AS r, SECTION AS sn, QUALIFIED AS q, FACULTY
AS f
WHERE  f.FacultyID =  q. FacultyID
AND    r.SectionNo =  sn. SectionNo
AND    q.CourseID =  sn. CourseID
AND    f.FacultyName = ‘Collins’
AND    sn.Semester = ‘I-2018’;
```

(Or)

(We can use join statement):

```
SELECT COUNT(r.StudentID)
FROM REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN QUALIFIED q
NATURAL JOIN FACULTY f
WHERE f.FacultyName = 'Collins' AND sn.Semester = 'I-2018';
```

(Note: We can use natural join here as the it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

(Or)

(We can use sub query by using in statement):

```
SELECT Count(StudentID)
FROM REGISTRATION
WHERE SectionNo
IN (SELECT SectionNo FROM SECTION WHERE CourseID IN (SELECT
CourseID FROM QUALIFIED WHERE FacultyID = (SELECT FacultyID FROM
FACULTY WHERE FacultyName = 'Collins'))
AND Semester = 'I-2018');
```

e.

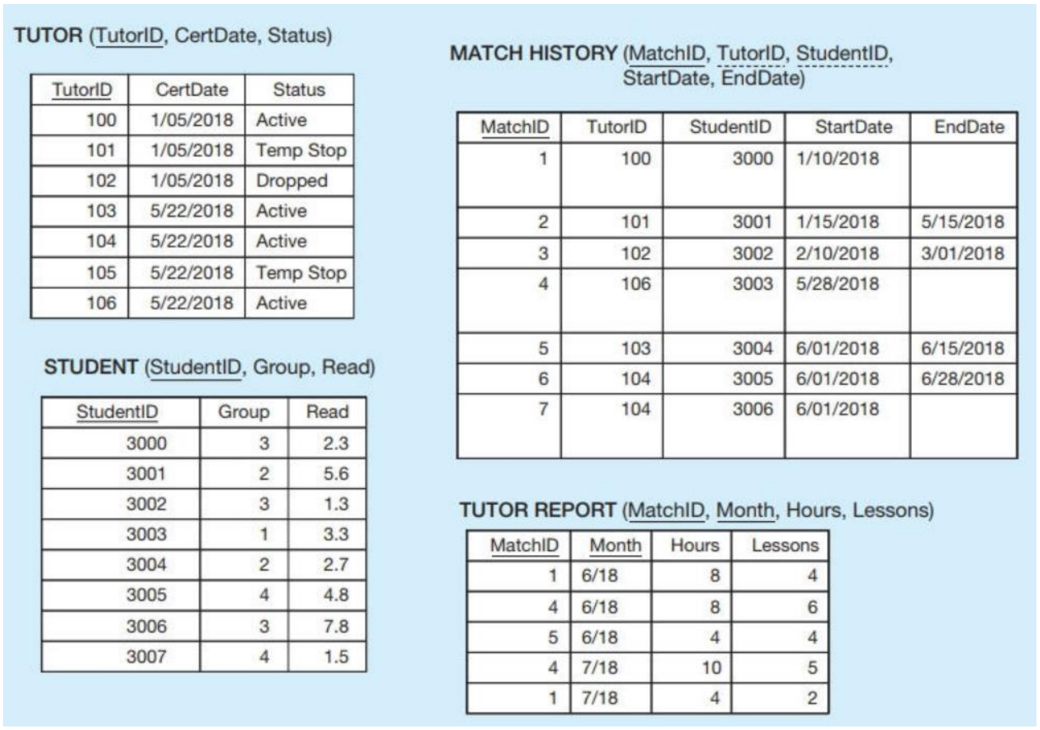
```
SELECT COURSE.CourseName
FROM COURSE, QUALIFIED
WHERE COURSE.CourseID = QUALIFIED.CourseID
GROUP BY CourseID
```

```
HAVING COUNT (FacultyID) >=2;
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT c.CourseName
FROM COURSE AS c, QUALIFIED AS  q
WHERE c.CourseID = q.CourseID
GROUP BY CourseID
HAVING COUNT (FacultyID) >=2;
```

Q2. Provide the outputs by the following SQL statements [30]



TUTOR (TutorID, CertDate, Status)

TutorID	CertDate	Status
100	1/05/2018	Active
101	1/05/2018	Temp Stop
102	1/05/2018	Dropped
103	5/22/2018	Active
104	5/22/2018	Active
105	5/22/2018	Temp Stop
106	5/22/2018	Active

STUDENT (StudentID, Group, Read)

StudentID	Group	Read
3000	3	2.3
3001	2	5.6
3002	3	1.3
3003	1	3.3
3004	2	2.7
3005	4	4.8
3006	3	7.8
3007	4	1.5

MATCH HISTORY (MatchID, TutorID, StudentID, StartDate, EndDate)

MatchID	TutorID	StudentID	StartDate	EndDate
1	100	3000	1/10/2018	
2	101	3001	1/15/2018	5/15/2018
3	102	3002	2/10/2018	3/01/2018
4	106	3003	5/28/2018	
5	103	3004	6/01/2018	6/15/2018
6	104	3005	6/01/2018	6/28/2018
7	104	3006	6/01/2018	

TUTOR REPORT (MatchID, Month, Hours, Lessons)

MatchID	Month	Hours	Lessons
1	6/18	8	4
4	6/18	8	6
5	6/18	4	4
4	7/18	10	5
1	7/18	4	2

```
1. SELECT Tutor.TutorID, status, StudentID
   FROM Tutor Left Join MatchHistory
   ON Tutor.TutorID = MatchHistory.TutorID
   Where Tutor.TutorID = 104;
```

ANSWER: The output of the above query will be:

TutorID	Status	StudentID
104	Active	3005
104	Active	3006

```
2. SELECT Student.StudentID, Read
   From Student Left Join MatchHistory
   ON Student.StudentID = MatchHistory.StudentID
   Where TutorID is NULL
```

ANSWER: The output of the above query will be:

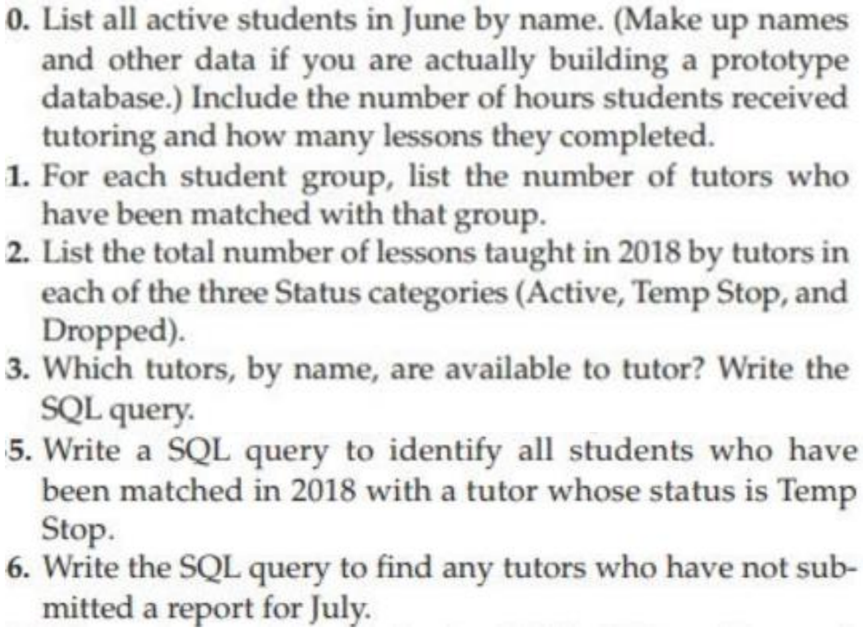
StudentID	Read
3007	1.5

```
3. SELECT Student.StudentID, Read, Tutor.TutorID, Status
   From student
   Left Join MatchHistory ON Student.StudentID = MatchHistory.StudentID
   Left Join Tutor ON Tutor.TutorID = MatchHistory.TutorID
   Where Student.StudentID > 3004;
```

ANSWER: The output of the above query will be:

StudentID	Read	TutorID	Status
3005	4.8	104	Active
3006	7.8	104	Active
3007	1.5	NULL	NULL

Q3. Write down SQL statements based on DB in Q2 [30]



0. List all active students in June by name. (Make up names and other data if you are actually building a prototype database.) Include the number of hours students received tutoring and how many lessons they completed.
1. For each student group, list the number of tutors who have been matched with that group.
2. List the total number of lessons taught in 2018 by tutors in each of the three Status categories (Active, Temp Stop, and Dropped).
3. Which tutors, by name, are available to tutor? Write the SQL query.
5. Write a SQL query to identify all students who have been matched in 2018 with a tutor whose status is Temp Stop.
6. Write the SQL query to find any tutors who have not submitted a report for July.

ANSWER:

0. As there is no ‘student name’ column in the student table, assume that we added ‘StudentName’ column in the student table. Then the query will be :

```
SELECT STUDENT.StudentID, STUDENT.StudentName, COUNT(TUTOR REPORT.Hours) AS Num_Of_hours, COUNT(TUTOR REPORT.Lessons) AS Num-Of_Lessons
FROM STUDENT, MATCH HISTORY, TUTOR REPORT
WHERE STUDENT.StudentID = MATCH HISTORY.StudentID
AND MATCH HISTORY.MatchID = TUTOR REPORT.MatchID
AND EXTRACT (MONTH FROM (MATCH HISTORY. EndDate) >= 6
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT s.StudentID, s.StudentName, COUNT(tr.Hours) AS Num_Of_hours,
COUNT(tr.Lessons) AS Num-Of_Lessons
FROM STUDENT AS s, MATCH HISTORY AS mh, TUTOR REPORT AS tr
WHERE s.StudentID = mh.StudentID
AND mh.MatchID = tr.MatchID
AND EXTRACT (MONTH FROM (mh. EndDate) >= 6
```

(Or)
(We can extract month in other way by changing the last line of the query):

```
AND tr.Month LIKE '6%';
```

(Or)
(We can use join statement):

```
SELECT s.StudentID, s.StudentName, COUNT(tr.Hours) AS Num_Of_hours,
COUNT(tr.Lessons) AS Num-Of_Lessons
FROM STUDENT AS s
INNER JOIN MATCH HISTORY AS mh ON s.StudentID = mh.StudentID
LEFT JOIN TUTOR REPORT AS tr ON mh.MatchID = tr.MatchID
WHERE mh.EndDate IS NULL
OR EXTRACT (MONTH FROM (mh. EndDate) >= 6
GROUP BY s.StudentID, s. StudentName;
```

1.

```
SELECT  STUDENT. Group, COUNT(MATCH HISTORY.TutorID) AS
Num_Of_Tutors
FROM STUDENT , MATCH HISTORY
WHERE  STUDENT. StudentID   =   MATCH HISTORY. StudentID
GROUP BY STUDENT.Group;
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT  s.StudentID , COUNT(mh.TutorID) AS Num_Of_Tutors
FROM STUDENT AS s , MATCH HISTORY AS mh
WHERE  s. StudentID   =   mh. StudentID
GROUP BY s.Group;
```

2.

```
SELECT COUNT (TUTOR REPORT. Lessons) AS Num_Of_Lessons,
TUTOR.TutorID, TUTOR.Status
FROM  TUTOR REPORT, TUTOR, MATCH HISTORY
WHERE TUTOR.TutorID = MATCH HISTORY. TutorID
AND TUTOR REPORT. MatchID =  MATCH HISTORY.MatchID
AND EXTRACT (YEAR FROM  (MATCH HISTORY. StartDate)) = 2018
OR EXTRACT (YEAR FROM (MATCH HISTORY.EndDate)) = 2018
GROUP BY TUTOR.Status
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT COUNT (tr. Lessons) AS Num_Of_Lessons  , t.TutorID,  t.Status
FROM  TUTOR REPORT AS tr, TUTOR AS t, MATCH HISTORY AS mh
WHERE t.TutorID = mh. TutorID
AND tr. MatchID =  mh.MatchID
AND EXTRACT (YEAR FROM  (mh. StartDate)) = 2018
OR EXTRACT (YEAR FROM (mh.EndDate)) = 2018
GROUP BY t.Status
```

(Or)
(We can use join statement):

```
SELECT t.Status, SUM(tr.Lessons) AS TotalLessons
FROM Tutor t
```

```
LEFT JOIN mh ON t.TutorID = mh.TutorID
LEFT JOIN tr ON mh.MatchID = tr.MatchID
WHERE EXTRACT(YEAR FROM TO_DATE(mh.StartDate, 'MM/DD/RRRR')) =
2018
GROUP BY t.Status;
```

3.

As there is no TutorName column in the Tutor table, assume that we added a new column called 'TutorName' to the Tutor table. Then the query will be:

```
SELECT  TutorID, Name, Status

FROM TUTOR

WHERE Status = 'Active';
```

5.

```
SELECT  MATCH HISTORY.StudentID
FROM  MATCH HISTORY, TUTOR
WHERE  MATCH HISTORY.TutodID  =  TUTOR.TutorID
AND  TUTOR .Status = 'Temp Stop'
AND  EXTRACT (YEAR FROM (TUTOR.CertDate)) = 2018;
```

(Or)

(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT  mh.StudentID
FROM  MATCH HISTORY AS mh , TUTOR AS t
WHERE  mh.TutodID  =  t.TutorID
AND  t.Status = 'Temp Stop'
AND  EXTRACT (YEAR FROM (t.CertDate)) = 2018;
```

6.

```
SELECT  MATCH HISTORY.TutorID
FROM  MATCH HISTORY, TUTOR REPORT
WHERE   MATCH HISTORY.MatchID =   TUTOR REPORT. MatchID
AND   TUTOR REPORT. Month  !=  '7/18' ;
```

(Or)

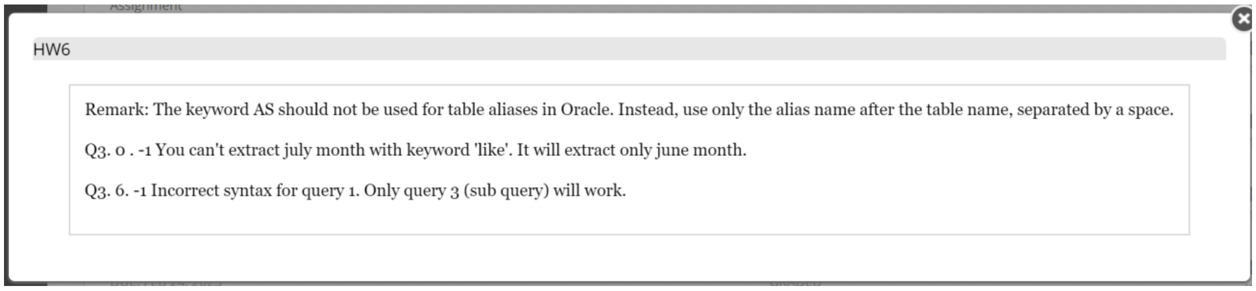
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT  mh.TutorID
FROM  MATCH HISTORY AS mh , TUTOR REPORT AS tr
WHERE   mh.MatchID =   tr. MatchID
AND   tr. Month  !=  '7/18' ;
```

(Or)
(using join and subquery)

```
SELECT TutorID, Name
FROM Tutor
WHERE Status = ‘Active’ AND TutorID NOT IN
(SELECT TutorID FROM MatchHistory mh
INNER JOIN TutorReport tr ON tr.MatchID = mh.MatchID
WHERE EXTRACT(MONTH FROM TO_DATE(tr.Month, ‘MM/DD’)) = 7);
```

FEEDBACK & SCORE : 98 but for all the 3 questions, don’t use ‘as’ for renaming table names as it don’t work in oracle 21c. so the 2nd way is wrong in this assignment. 1st and join approach is correct.



HW6

Remark: The keyword AS should not be used for table aliases in Oracle. Instead, use only the alias name after the table name, separated by a space.

Q3. 0 . -1 You can't extract july month with keyword 'like'. It will extract only june month.

Q3. 6. -1 Incorrect syntax for query 1. Only query 3 (sub query) will work.