Name	Nachiket Rao
Student ID	S3827657
Tutorial: **Day, Time , Location**	Tuesday, 10.30-12.30, 014.09.015
Tutor Name	Kenneth Gardiner

DATABASE CONCEPTS

Assignment 2

Question 1 – SQL

1.1 – Query explanation

This query fetches the details (names and institute name) of academics who are part of Computer Science department and have no interest in any field

1.2 – Correct SQL Query

The SQL query tries to implicitly join the same author table and additionally in doing so also does not specify proper alias names for the parameters (select Panum – which panum) , (Panum = A2.panum – panum on left hand side is ambiguous)

Correct Query –

```
SELECT panum,
       Count(acnum)
FROM   author
GROUP  BY panum
```

1.3 –

```
SELECT *
FROM   department
WHERE  descrip IS NOT NULL
```

1.4 –

```
SELECT *
FROM   paper
       NATURAL join author
WHERE  acnum = 100
```

1.5 –

```
SELECT acnum,
       givename,
       famname,
       Count(panum)
FROM   academic
       NATURAL join author
GROUP  BY acnum,
          givename,
          famname
ORDER  BY acnum ASC
```

1.6 –

```
SELECT id,
       Count(acnum)
FROM   interest
       NATURAL join field
WHERE  id LIKE '_.1%'
```

```
GROUP  BY id
ORDER  BY id ASC
```

1.7 –

```
SELECT DISTINCT deptnum,
                deptname,
                instname
FROM   department
       NATURAL join academic
WHERE  acnum NOT IN (SELECT DISTINCT acnum
                     FROM   interest)
```

1.8 –

```
SELECT acnum,
       famname,
       givename,
       deptnum,
       descrip
FROM   academic
       NATURAL join department
WHERE  famname LIKE 'C%'
ORDER  BY famname,
          givename
```

1.9 –

```
SELECT fieldnum,
       title,
       Count(acnum) AS "NO. ACADEMICS INTERESTED"
FROM   field
       NATURAL join interest
GROUP  BY fieldnum,
          title
ORDER  BY fieldnum ASC
```

1.10 –

```
SELECT deptname,
       instname,
       Count(academic.acnum) AS "TotalAcademics"
FROM   department
       inner join academic
               ON academic.deptnum = department.deptnum
GROUP  BY deptname,
          instname
HAVING Count(academic.acnum) >= 10
```

1.11 –

```
SELECT DISTINCT deptnum
FROM   department
       NATURAL join academic
WHERE  ( postcode >= 3000
         AND postcode <= 3999 )
       AND Lower(title) NOT LIKE ( '%prof%' )
UNION ALL
(SELECT deptnum
 FROM   department
 WHERE  ( postcode >= 3000
          AND postcode <= 3999 )
 MINUS
 SELECT deptnum
 FROM   academic);
```

1.12 –

```
SELECT academic.deptnum,
       department.instname,
       department.deptname,
       Count(author.panum)
FROM   department
       inner join academic
               ON academic.deptnum = department.deptnum
       inner join author
               ON author.acnum = academic.acnum
GROUP  BY department.deptname,
          department.instname,
          academic.deptnum
HAVING Count(author.panum) >= 10
ORDER  BY academic.deptnum,
          department.deptname
```

1.13 –

```
SELECT deptnum,
       deptname
FROM   department
       NATURAL join academic
WHERE  acnum NOT IN (SELECT DISTINCT acnum
                     FROM   author)
```

1.14 –

```
SELECT panum
FROM   author
WHERE  EXISTS (SELECT DISTINCT acnum
               FROM   interest
               WHERE  Lower(descrip) LIKE '%data%'
                      AND interest.acnum = author.acnum);
```

1.15 –

```
SELECT fieldnum,
       id,
       title,
       Count(acnum)
FROM   field
       NATURAL join interest
GROUP  BY fieldnum,
          id,
          title
HAVING Count(acnum) = (SELECT Max(occ)
                       FROM   (SELECT fieldnum,
                                      id,
                                      title,
                                      Count(acnum) AS occ
                               FROM   field
                                      NATURAL join interest
                               GROUP  BY fieldnum,
                                         id,
                                         title
                               ORDER  BY Count(acnum) DESC))
```

Question 2 – Relational Model

2.1 – Give all FD's

FD 1: deptID → deptName, Manager

FD 2: empID → empName, deptID, email

FD 3: empID, projID → role

FD 4: projID → startYear, deptID

FD 5: projID, evalDate → grade

2.2 – Closure

$\{empID, projID\}^+$ = {empID, projID, role, empName, deptID, email, startYear, deptName, manager}

$\{deptID\}^+$ = {deptID, deptName, manager}

2.3 – Specify keys

Department (<u>deptID</u>, deptName, manager)

Employee (<u>empID</u>, empName, deptID*, email

Project (<u>projID</u>, startYear, deptID)

EmpProj (<u>empID*, projID</u>*, role)

Evaluation (<u>projID*, manager, evalDate*</u>, grade)

2.4 – Discuss normal form

The given relation Evaluation (projID, manager, evalDate, grade) is in 1NF.

Reason – The given relation has one candidate key {projID, EvalDate} and two FD's i.e

ProjID → Manager & ProjID, EvalDate → Grade. Out of these, the relation has one partial dependency which is ProjID → Manager (non-prime attribute Manager is dependent on a prime attribute ProjID). This violates the condition the definition of 2NF which requires a relation to have no partial dependency and as a result also cannot be 3NF and BCNF.

Question 3 – Normalisation

3.1 – Minimal Basis FD

Step 1 - Splitting FD's with multiple attributes on the right

FD 1: docID → docName (No change, only one attribute on the right)

FD 2: patID → patName, patDOB

a) patID → patName
b) patID → patDOB

FD 3: patID, appDate, appTime → docID, roomNo

a) patID, appDate, appTime → docID
b) patID, appDate, appTime → roomNo

FD 4: appDate, docID → roomNo (No change, only one attribute on the right)

FD 5: patID, appDate → appTime, roomNo, docID

a) patID, appDate → appTime
b) patID, appDate → roomNo
c) patID, appDate → docID

Step 2 – Removing redundant FD's

~~patID, appDate, appTime → roomNo~~ (Redundant attribute appTime on leftside)

~~patID, appDate, appTime → docID~~ (Redundant attribute appTime on leftside)

Final Minimal Basis FD –

docID → docName

patID → patName

patID → patDOB

appDate, docID → roomNo

patID, appDate → appTime

patID, appDate → roomNo

patID, appDate → docID

3.2 – Give all Candidate Keys

Candidate Key - {patID, appDate}

Explanation –

- Closure of {patID, appDate} gives all the attributes of the relation APP

  $\{patID, appDate\}^+ = \{patID, appDate, patDOB, appTime, roomNo, docID, docName\}$

- Any subset of {patID, appDate} cannot determine the relation APP
  $\{patID\}^+ = \{patID, patDOB\}$
  $\{appDate\}^+ = \{appDate\}$
- Superset of {patID, appDate} cannot minimally determine the relation APP

3.3 – BCNF form

Step 1 – Construct minimal basis FD

Refer question 3.1

Step 2 – Find candidate keys

Refer question 3.2

Step 3 – Create relation from all minimal FD

R1 (doc id, patName)

R2 (patID, patName)

R3 (patID, patDOB)

R4 (appDate, docID, roomNo)

R5 (patID, appDate, appTime)

R6 (patID, appDate, roomNo)

R7 (patID, appTime, docID)

Step 4 – Combining Relations to get final BCNF form

APP1 (<u>docID</u>, docName)

APP2 (<u>patID</u>, patName, patDOB)

APP3 (<u>patID*, appDate</u>, appTime, docID*, roomNo)

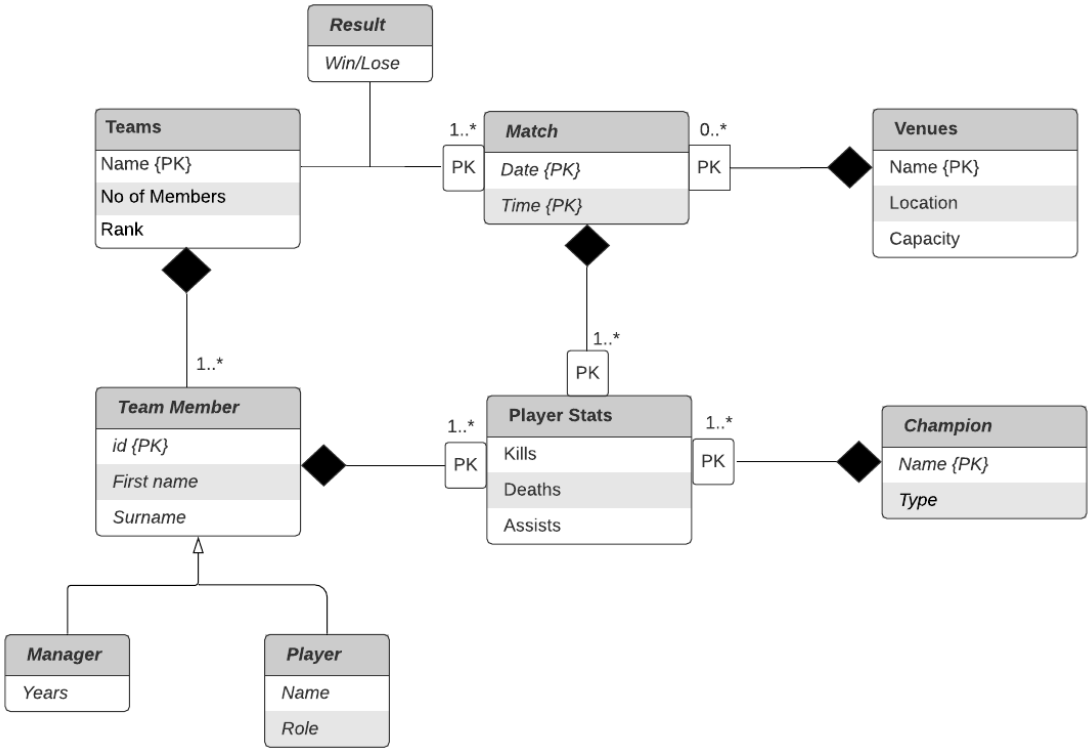
APP4 (<u>appDate, docID*</u>, roomNo)

Question 4 – ER Diagram

This diagram was made using Lucidchart software [1]

Assumptions:

- Teams are uniquely identified by their names and is an entity
- Match is a weakly identified entity keeping track of matches while result is an association class keeping track of the outcome of matches



Unexplained Constraints/Ambiguities in Question description:

- It is unclear whether a manager can be a player and vice versa.

References:

[1] Lucidchart.com. 2020. [online] Available at: <https://www.lucidchart.com/> [Accessed 2 April 2020].

Question 5 – ER to Relational

Class (cno*, grpNo, eno*, day, time, roomNo, type)

Course (cno, Title)

Student (sno, givename, surname, DOB, addr)

Enroll (sno*, cno*, grade)

Takes (sno*, cno*, grpNo*)

Staff (eno, givename, surname)

Tutor (t_eno, givename, surname, contract)