1a)

SELECT \*

FROM Patient

WHERE MONTH(Date\_admitted) = 6 AND YEAR(Date\_admitted) = 2017

ORDER BY Date\_admitted DESC;

1b)

SELECT WARD.Ward\_name, consultant.Consultant\_name, COUNT(patient\_ward.Patient\_ID) AS COUNT

FROM patient\_ward

JOIN WARD ON patient\_ward.Ward\_no = WARD.Ward\_no

JOIN consultant ON patient\_ward.Consultant\_no = consultant.Consultant\_no

GROUP BY patient\_ward.Ward\_no, WARD.Ward\_name, consultant.Consultant\_name

ORDER BY patient\_ward.Ward\_no, COUNT(patient\_ward.Patient\_ID) DESC;

1c)

SELECT MONTH(DATE\_ADMITTED) AS MONTH, COUNT(PATIENT\_ID) AS "NUMBER OF PATIENTS"

FROM Patient

GROUP BY MONTH(DATE\_ADMITTED);

1d)

SELECT \*

FROM WARD

WHERE WARD.Ward\_no NOT IN (SELECT patient\_ward.Ward\_no FROM patient\_ward);

1e)

SELECT WARD.Ward\_no, WARD.Ward\_name, Staff.Staff\_name, X.[Number of Staff], X.[Number of Patients], X.[Number of Consultants]

FROM WARD

LEFT JOIN Staff ON WARD.In\_charge = Staff.Staff\_no

LEFT JOIN

(

SELECT staff\_duty.Ward\_no,

COUNT(DISTINCT staff\_duty.Staff\_no) AS "Number of Staff",

COUNT(DISTINCT patient\_ward.Patient\_ID) AS "Number of Patients",

COUNT(DISTINCT patient\_ward.Consultant\_no) AS "Number of Consultants"

FROM staff\_duty

LEFT JOIN patient\_ward ON staff\_duty.Ward\_no = patient\_ward.Ward\_no

GROUP BY staff\_duty.Ward\_no

) AS X ON WARD.Ward\_no = X.Ward\_no

ORDER BY X.[Number of Consultants] DESC, X.[Number of Patients]

Qn2

LecturerCourse (lecturerId, lecturerName, lecturerEmail, lecturerGender, courseCode, courseName, startDate, startTime, weeklyHours, noofWeeks, learnerId, learnerName, learnerEmail)

**Functional dependencies**

lecturerId 🡪 (lecturerName, lecturerEmail, lecturerGender)

courseCode 🡪 (courseName, weeklyHours, noofWeeks)

learnerId 🡪 (learnerName, learnerEmail)

courseCode 🡪 (startDate, startTime)

**Multi-valued dependencies**

courseCode 🡪🡪 (startDate, startTime)

learnerId 🡪🡪 (courseCode, startDate, startTime)

lecturerId 🡪🡪 (courseCode, startDate, startTime)

**Candidate key**

(courseCode, startDate, startTime)

**Assemble**

Lecturer (lecturerId, lecturerName, lecturerEmail, lecturerGender)

Courses (courseCode, courseName, weeklyHours, noofWeeks)

Learner (learnerId, learnerName, learnerEmail)

CourseSchedule (courseCode, startDate, startTime)

LearnerRecord (learnerId, courseCode, startDate, startTime)

LecturerSchedule (lecturerId, courseCode, startDate, startTime)

Step 1

Step 2

Step 3

**Functional dependency**: lecturerId 🡪 (lecturerName, lecturerEmail, lecturerGender)

Move the column of the functional dependency into a new relation

Lecturer (lecturerId, lecturerName, lecturerEmail, lecturerGender)

Make the determinant of that functional dependency the primary key of the new relation

Lecturer (**lecturerId**, lecturerName, lecturerEmail, lecturerGender)

Leave a copy of the determinant as a foreign key in the original relation

LecturerCourse (lecturerId, ~~lecturerName, lecturerEmail, lecturerGender~~, courseCode, courseName, startDate, startTime, weeklyHours, noofWeeks, learnerId, learnerName, learnerEmail)

Create a referential integrity constraint between the original relation and the new relation

Where LecturerCourse.lecturerId must exist in Lecturer.lecturerId

**Functional dependency**: courseCode 🡪 (courseName, weeklyHours, noofWeeks)

Move the column of the functional dependency into a new relation

Courses (courseCode, courseName, weeklyHours, noofWeeks)

Make the determinant of that functional dependency the primary key of the new relation

Courses (**courseCode**, courseName, weeklyHours, noofWeeks)

Leave a copy of the determinant as a foreign key in the original relation

LecturerCourse (lecturerId, ~~lecturerName, lecturerEmail, lecturerGender~~, courseCode, ~~courseName~~, startDate, startTime, ~~weeklyHours, noofWeeks~~, learnerId, learnerName, learnerEmail)

Create a referential integrity constraint between the original relation and the new relation

Where LecturerCourse.courseCode must exists in Courses.courseCode

**Functional dependency**: learnerId 🡪 (learnerName, learnerEmail)

Move the column of the functional dependency into a new relation

Learner (learnerId, learnerName, learnerEmail)

Make the determinant of that functional dependency the primary key of the new relation

Learner (**learnerId**, learnerName, learnerEmail)

Leave a copy of the determinant as a foreign key in the original relation

LecturerCourse (lecturerId, ~~lecturerName, lecturerEmail, lecturerGender~~, courseCode, ~~courseName~~, startDate, startTime, ~~weeklyHours, noofWeeks~~, learnerId, ~~learnerName, learnerEmail~~)

Create a referential integrity constraint between the original relation and the new relation

Where LecturerCourse.learnerId must exists in Learner.learnerId

Put multi-valued dependencies in their own relations

LecturerCourse (lecturerId, courseCode, startDate, startTime, learnerId)

LecturerCourse (lecturerId , courseCode, startDate, startTime)

LearnerEnrollment (learnerId, courseCode)

Functional dependency:

Move the column of the functional dependency into a new relation

Make the determinant of that functional dependency the primary key of the new relation

Leave a copy of the determinant as a foreign key in the original relation

Create a referential integrity constraint between the original relation and the new relation