CE205 Databases and Information Retrieval

Scenario Modelling and Database Implementation

1. Functional Dependencies (FD) in the relation

1.1 Relation with Sample Data [5%]

studentID	studentName	moduleID	moduleTitle	staffID	staffName	moderatorID	assessmentID	AssessmentType	marksObtained	AssessmentPercentage	Deadline
STU001	Mitchell	CS08	Cyber Security	T10	James	T05	A04	Project	29	40	28/10/25
STU001	Mitchell	CS08	Cyber Security	T10	James	T05	A03	Final Exam	41	60	15/11/25
STU001	Mitchell	CS12	Databases	T11	Basir oskoei	T12	A01	Assignment	68	30	20/10/25
STU001	Mitchell	CS12	Databases	T11	Basir oskoei	T12	A02	Progress Test	62	20	05/11/25
STU001	Mitchell	CS12	Databases	T11	Basir oskoei	T12	A03	Final Exam	55	50	10/12/25
STU002	michael jordan	CS08	Cyber Security	T10	James	T05	A04	Project	72	40	28/10/25
STU002	michael jordan	CS08	Cyber Security	T10	James	T05	A03	Final Exam	64	60	15/11/25
STU002	michael jordan	CS15	Application Programming	T12	Sam	T11	A01	Assignment	81	30	18/10/25
STU002	michael jordan	CS15	Application Programming	T12	Sam	T11	A02	Progress Test	74	20	03/11/25
STU002	michael jordan	CS15	Application Programming	T12	Sam	T11	A03	Final Exam	67	50	08/12/25

1.2 Functional Dependencies (FDs) [10%]

studentID - studentName: Each student has their own ID identifying them

staffID - staffName: Each staff has their own ID identifying them

moderatorID -> staffName: Each Moderator is also another staff member with their own ID

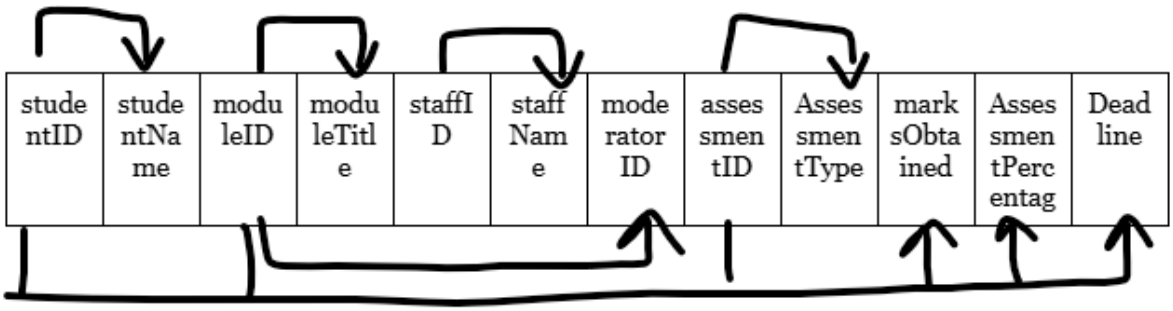
moduleID - moduleTitle, staffID, moderatorID: Each module has their own Title name, one staff member, one Moderator

(moduleID, assessmentID) determines AssessmentType, AssessmentPercentage, Deadline: Each of the modules has their own exam that belongs to them

(studentID, assessmentID) - marksObtained: Each of the marks of the student depends on their assessment

StudentID, ModuleID, AssessmentID ->

1.3 Graphical representation of FDS [5%]



2. Primary key of the relation

2.1 Primary key of the relation using the closure operator '+' [5%]

studentID alone will give us studentName only

moduleID alone will give us moduleTitle, staffID, moderatorID, but not student or marks

assessmentID alone will give us assessment details but not the student who took it

So combining them all will give us a single student's mark in a specific module assessment so our Primary key is studentID, moduleID, assessmentID

2.2 Prime and non-prime attributes [5%]

Prime Attributes: studentID, moduleID, assessmentID (they form the key)

Non-Prime Attributes: studentName, moduleTitle, staffID, staffName, moderatorID, AssessmentType, AssessmentPercentage, Deadline, marksObtained

3. Types of Functional Dependencies (FD) in the Relation

3.1 Partial FD(s) and justification [4%]

Partial FD(s):

studentID: studentName
moduleID: moduleTitle, staffID, moderatorID
(moduleID, assessmentID): AssessmentType, AssessmentPercentage, Deadline

Reason:

These attributes only depend on part of the composite key not the whole key

3.2 Transitive FD(s) and justification [3%]

Transitive FD(s):

moduleID: staffID: staffName
moduleID: moderatorID: staffName

Reason: staffName depends on moduleID indirectly through staffID or moderatorID

3.3 Full key FD(s) and justification [3%]

Full key FD(s):

(studentID, moduleID, assessmentID): marksObtained

Reason: The student's mark depends on all three key attributes together not just part of the key

4. Normalisation to 2NF [15%]

4.1 Process for 2nd normal form [10%]

primary key: (studentID, moduleID, assessmentID)

partial FDs:

studentID: studentName
moduleID: moduleTitle, staffID, moderatorID
(moduleID, assessmentID): AssessmentType, AssessmentPercentage, Deadline

To remove these, the relation is decomposed into the following tables:

STUDENT(studentID, studentName)
MODULE(moduleID, moduleTitle, staffID, moderatorID)
ASSESSMENT(moduleID, assessmentID, AssessmentType, AssessmentPercentage, Deadline)
MARKS(studentID, moduleID, assessmentID, marksObtained)

Now every nonkey attribute depends on a whole key not part of one.

4.2 Primary key(s) and foreign key(s) in each decomposed sub-relations [5%]

Table	Primary Key	Foreign Keys
STUDENT	studentID	N/A
MODULE	moduleID	N/A
ASSESSMENT	moduleID, assessmentID	moduleID is a foreign key referencing the MODULE table
MARKS	StudentID, moduleID, assessmentID	studentID is a foreign key referencing the STUDENT table; moduleID is a foreign key referencing the MODULE table; (moduleID, assessmentID) is a foreign key referencing the ASSESSMENT table

5. Normalisation to 3NF

5.1 Process for 3rd normal form [10%]

In the 2NF design, the relation MODULE(moduleID, moduleTitle, staffID, moderatorID, staffName) has a transitive dependency because the staffName depends on the staffID and

moderatorID, not directly on moduleID. To delete this issue, the STAFF table was made to store the details in a separate table; now the MODULE table references STAFF through staffID and moderatorID.
Therefore all transitive dependencies have been deleted and all of the nonkey attributes depend on the key.

Final relations after converting to 3NF:

1. STUDENT(studentID, studentName)
2. STAFF(staffID, staffName)
3. MODULE(moduleID, moduleTitle, staffID, moderatorID)
4. ASSESSMENT(moduleID, assessmentID, AssessmentType, AssessmentPercentage, Deadline)
5. MARKS(studentID, moduleID, assessmentID, marksObtained)

5.2 Primary key(s) and foreign key(s) in each decomposed sub-relations [5%]

Table	Primary Key	Foreign Key
STUDENT	studentID	N/A
STAFF	staffID	N/A
MODULE	moduleID	staffID and moderatorID
ASSESSMENT	moduleID, assessmentID	moduleID
MARKS	studentID, moduleID, assessmentID	studentID references STUDENT, moduleID references MODULE, and (moduleID, assessmentID) references ASSESSMENT

6. Final Design of the Database [5%]

List all decomposed relations clearly, including their keys *(this section is very crucial, and complete it carefully)*

1. **STUDENT(studentID, studentName)**
   Primary Key: studentID

2. **STAFF(staffID, staffName)**
   Primary Key: staffID

3. **MODULE(moduleID, moduleTitle, staffID, moderatorID)**
   Primary Key: moduleID
   Foreign Keys: staffID and moderatorID reference STAFF(staffID)

4. **ASSESSMENT(moduleID, assessmentID, AssessmentType, AssessmentPercentage, Deadline)**
   Primary Key: (moduleID, assessmentID)
   Foreign Key: moduleID references MODULE(moduleID)

5. **MARKS(studentID, moduleID, assessmentID, marksObtained)**
   Primary Key: (studentID, moduleID, assessmentID)

6. **Foreign Keys:**

   a. studentID references STUDENT(studentID)

   b. moduleID references MODULE(moduleID)

   c. (moduleID, assessmentID) references ASSESSMENT(moduleID, assessmentID)

7. Implementation using MySQL Workbench

7.1 Write code for each sub-relation in MySQL **[10%]** ***(provide .sql file)***

7.2 Write code to insert data in each sub-relation **[5%]** ***(provide .sql file)***

7.3 Write any of the following THREE queries in MySQL to retrieve information from the implemented data **[9%]** *(include screenshots in the report to show the output of each query **[6%])***

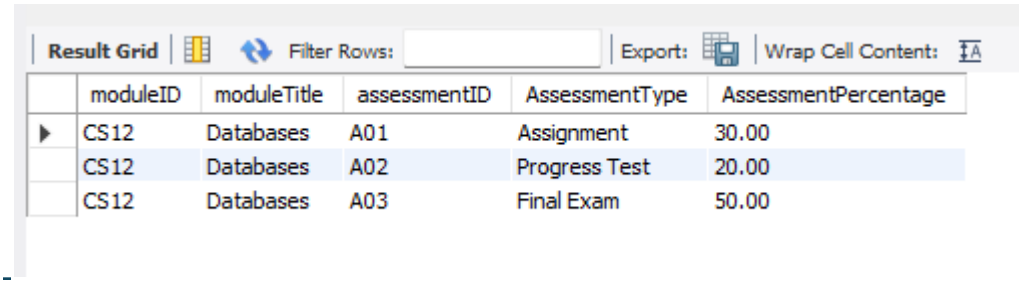
Queries

- List all assessments of a module (of your choice) and their respective weight percentages.
- Display all modules taught by a staff member (of your choice)
- List students who passed modules with their overall percentage (≥ 40%)
- Shows the number of modules taught by each staff member.
- Display which students are registered in which modules, showing student details and module titles.

Code for selected query 1:

```
SELECT
 a.moduleID,
 m.moduleTitle,
 a.assessmentID,
 a.AssessmentType,
 a.AssessmentPercentage
FROM ASSESSMENT a
JOIN MODULE m ON m.moduleID = a.moduleID
WHERE a.moduleID = 'CS12'
ORDER BY a.assessmentID;
```

Output of query 1:



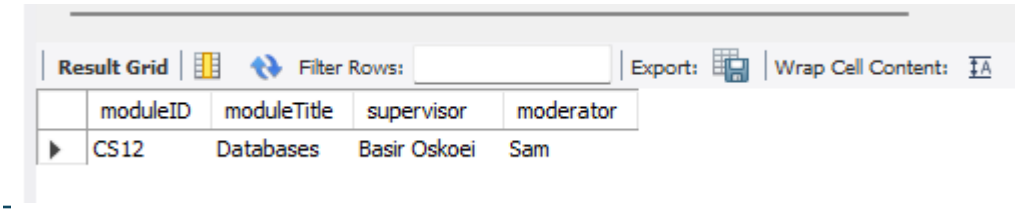
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

moduleID	moduleTitle	assessmentID	AssessmentType	AssessmentPercentage
CS12	Databases	A01	Assignment	30.00
CS12	Databases	A02	Progress Test	20.00
CS12	Databases	A03	Final Exam	50.00

Code for selected query 2:

```
SELECT
 m.moduleID,
 m.moduleTitle,
 s.staffName  AS supervisor,
 s2.staffName AS moderator
FROM MODULE m
JOIN STAFF s  ON s.staffID  = m.staffID
JOIN STAFF s2 ON s2.staffID = m.moderatorID
WHERE m.staffID = 'T11'
ORDER BY m.moduleID;
```

Output of query 2:

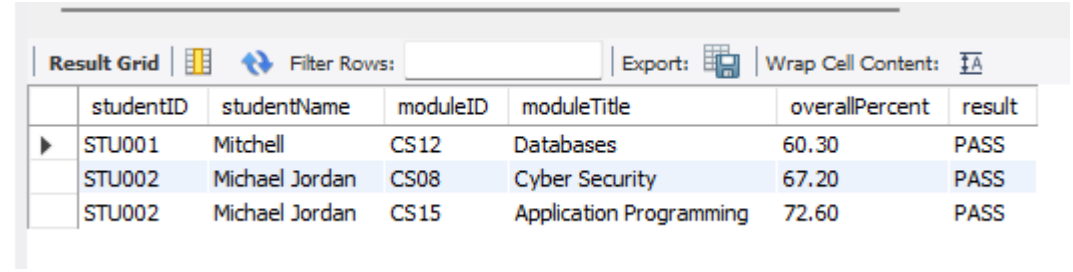


moduleID	moduleTitle	supervisor	moderator
CS12	Databases	Basir Oskoei	Sam

Code for selected query 3:

```
 SELECT
  mk.studentID,
  st.studentName,
  mk.moduleID,
  mo.moduleTitle,
  ROUND(SUM(mk.marksObtained * a.AssessmentPercentage) / 100, 2) AS overallPercent,
  CASE WHEN SUM(mk.marksObtained * a.AssessmentPercentage) / 100 >= 40
      THEN 'PASS' ELSE 'FAIL' END AS result
FROM MARKS mk
JOIN ASSESSMENT a
  ON a.moduleID = mk.moduleID AND a.assessmentID = mk.assessmentID
JOIN STUDENT st ON st.studentID = mk.studentID
JOIN MODULE  mo ON mo.moduleID  = mk.moduleID
GROUP BY mk.studentID, st.studentName, mk.moduleID, mo.moduleTitle
HAVING overallPercent >= 40
ORDER BY mk.studentID, mk.moduleID;
```

Output of query 3:



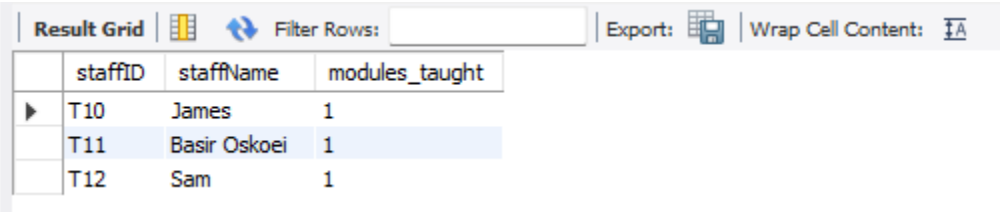
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

studentID	studentName	moduleID	moduleTitle	overallPercent	result
STU001	Mitchell	CS12	Databases	60.30	PASS
STU002	Michael Jordan	CS08	Cyber Security	67.20	PASS
STU002	Michael Jordan	CS15	Application Programming	72.60	PASS

Code for selected query 4:

```
SELECT
  s.staffID,
  s.staffName,
  COUNT(*) AS modules_taught
FROM MODULE m
JOIN STAFF s ON s.staffID = m.staffID
GROUP BY s.staffID, s.staffName
ORDER BY modules_taught DESC, s.staffID;
```

Output of query 4:



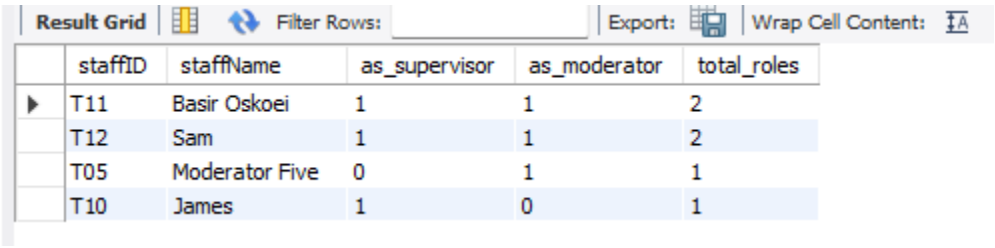
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

staffID	staffName	modules_taught
T10	James	1
T11	Basir Oskoei	1
T12	Sam	1

Code for selected query 5:

```
SELECT
  s.staffID,
  s.staffName,
  SUM(m.staffID = s.staffID)      AS as_supervisor,
  SUM(m.moderatorID = s.staffID)  AS as_moderator,
  COUNT(*) AS total_roles
FROM MODULE m
JOIN STAFF s ON s.staffID IN (m.staffID, m.moderatorID)
GROUP BY s.staffID, s.staffName
ORDER BY total_roles DESC, s.staffID;
```

Output of query 5:



staffID	staffName	as_supervisor	as_moderator	total_roles
T11	Basir Oskoei	1	1	2
T12	Sam	1	1	2
T05	Moderator Five	0	1	1
T10	James	1	0	1