1. Examine the dependency diagram carefully and answer the questions below.

a. Identify the primary key of the above dependency diagram. What is the name given to this type of primary key? [2 marks]

(A, D, G) is the primary key. This type of key is called a composite key.

b. What type of dependency is D→E and A→B,C? [2 marks]

D→E is partial dependency
A→B,C is partial dependency

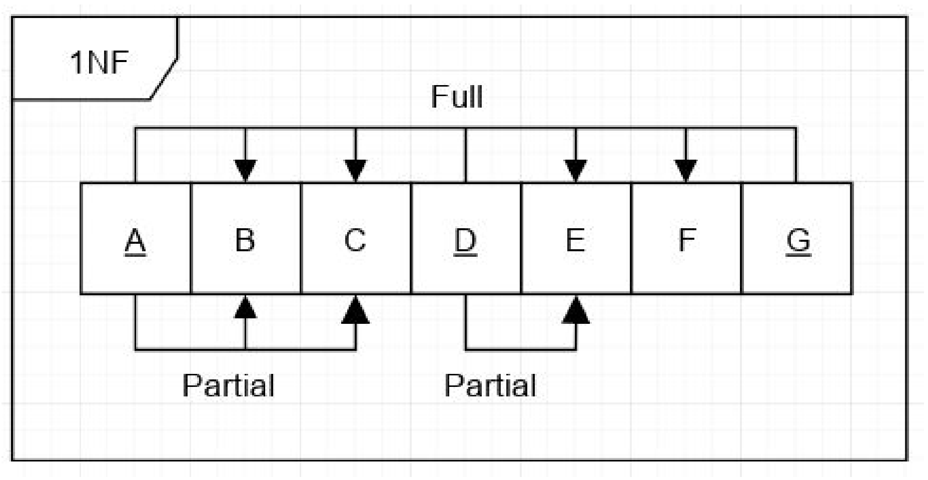
c. Can (A,D,G) determine F in the above diagram? Justify your answer. [2 marks]

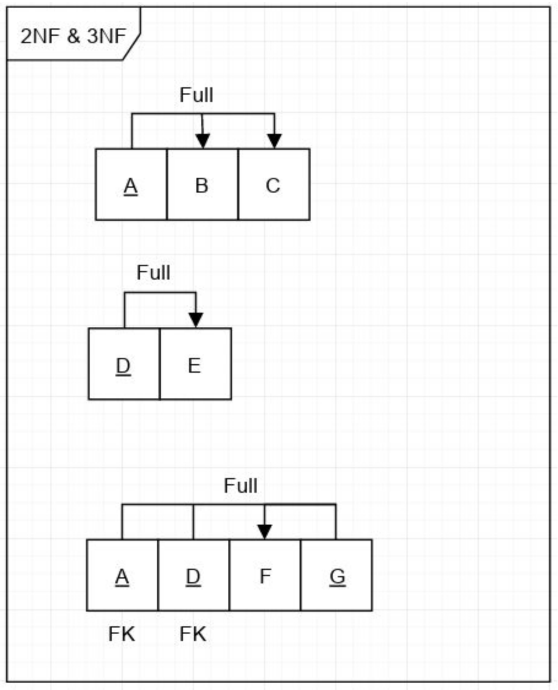
Yes because A,D,G→F is a full dependency. A,D,G is the primary key that determines attribute F.

d. Name two key attributes other than "A". [2 marks]

B and G

e. Normalise the above dependency table to 3rd normal form. You must show the progress from 1NF to 2NF, and then to 3NF. Indicate all the primary keys and foreign keys in the normalised tables. [8 marks]





f. Write the DDL to create the final 3NF tables. For primary keys use 'Integer' data type, and for non-key attributes use 'Text' data type. Use TABLE1, TABLE2, and TABLE3 etc. for the table name. [4 marks]

```
CREATE TABLE TABLE1(
        A INTEGER PRIMARY KEY NOT NULL,
        B TEXT,
        C TEXT
);

CREATE TABLE TABLE2(
        D INTEGER PRIMARY KEY NOT NULL,
        E TEXT
);

CREATE TABLE TABLE3(
        A INTEGER NOT NULL,
        D INTEGER NOT NULL,
        G INTEGER NOT NULL,
        PRIMARY KEY (A, D, G),
        F TEXT,
        FOREIGN KEY (`A`) REFERENCES `TABLE1` (`A`),
        FOREIGN KEY (`D`) REFERENCES `TABLE2` (`D`)
);
```

2. Given the constraint that a student can only register for ONE degree programme, answer the questions below

a. From the student record form above, identify the attributes likely to form a repeated group. [2 marks]

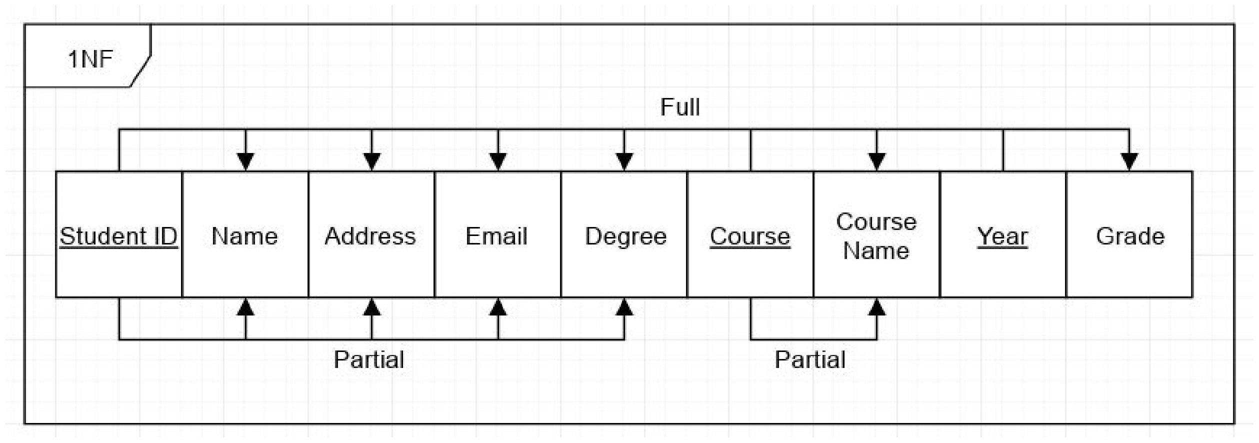
   Inside of course summary; Code, Year, Course Name, and Grade. These would be likely attributes to form a repeated group.

b. Transform the form into a table, then identify and write the primary key for the table. [4 marks]

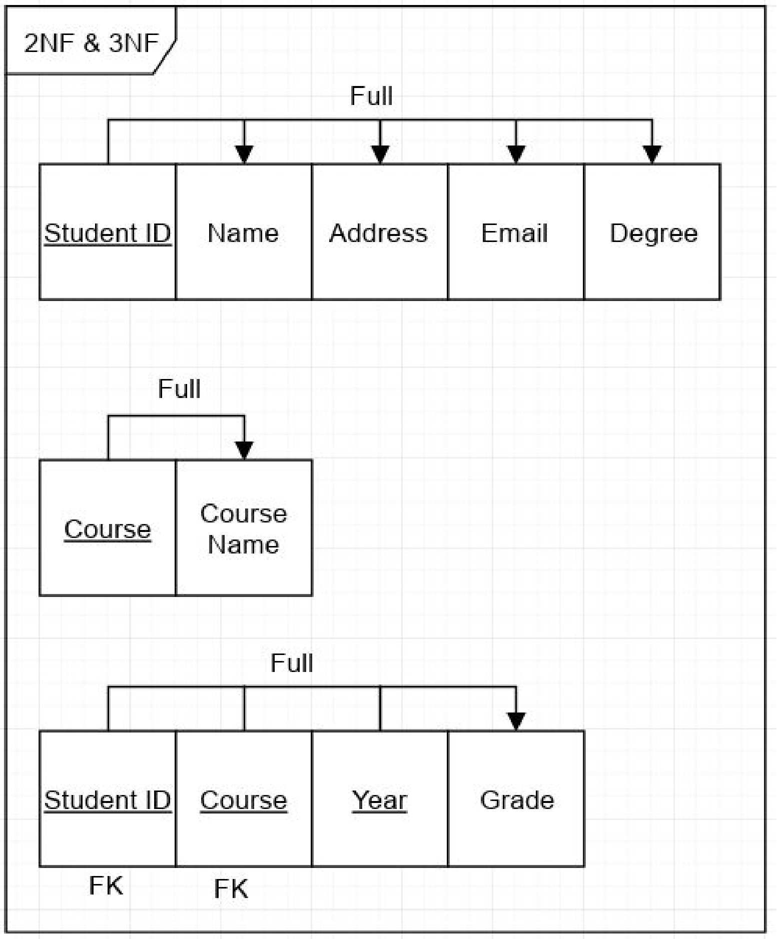
Student ID	**Name**	**Address**	**Email**	**Degree**	**Course**	**Year**	**Course Name**	**Grade**
12345678	James Bond	10 Downing Street, Wellington	jamesbond @gmail.com	Bachelor of Computer Science	INFO101	2013	Fundamentals of IS	F
12345678	James Bond	10 Downing Street, Wellington	jamesbond @gmail.com	Bachelor of Computer Science	FCOM100	2013	NZ Business Enviro	C
12345678	James Bond	10 Downing Street, Wellington	jamesbond @gmail.com	Bachelor of Computer Science	ELCM222	2015	E-business Strategy	B+
12345678	James Bond	10 Downing Street, Wellington	jamesbond @gmail.com	Bachelor of Computer Science	INFO101	2014	Fundamentals of IS	A
12345678	James Bond	10 Downing Street, Wellington	jamesbond @gmail.com	Bachelor of Computer Science	ELCM203	2014	Internet Apps	C

(Student ID, Course, Year) is the **primary key** of this table, which is a composite key.

c. Draw all the dependencies and identify the type of dependency for the 1NF table. [6 marks]



d. Normalise the 1NF table into 3rd normal form. You must show the progress from 1NF to 2NF, and then to 3NF. [8 marks]



e. Indicate all the primary keys and foreign keys in the normalised tables. [4 marks]

   Let us name these tables from top down order: StudentInfo, CourseInfo, GradeInfo.

   StudentInfo primary key is Student ID.
   CourseInfo primary key is Course.
   GradeInfo primary key is (Student ID,Course,Year) and foreign keys are Student ID and Course.

f. Write the DDL to create the normalised tables in a correct sequence. Use the table names StudentInfo, GradeInfo, and CourseInfo to represent the normalised tables. [6 marks]

```
CREATE TABLE StudentInfo(
        StudentID INTEGER PRIMARY KEY NOT NULL,
        Name TEXT,
        Address TEXT,
        Email TEXT,
        Degree TEXT
);

CREATE TABLE CourseInfo(
        Course TEXT PRIMARY KEY NOT NULL ,
        CourseName TEXT
);

CREATE TABLE GradeInfo(
        StudentID INTEGER NOT NULL,
        Course TEXT NOT NULL,
        Year INTEGER NOT NULL,
        PRIMARY KEY (StudentID Course, Year),
        Grade TEXT,
        FOREIGN KEY (`StudentID`) REFERENCES `StudentInfo` (`StudentID`),
        FOREIGN KEY (`Course`) REFERENCES `CourseInfo` (`Course`)
);
```

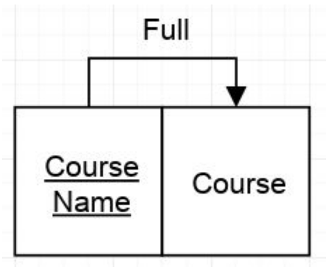
g. Write the INSERT SQL in a correct sequence to insert all the data shown in the sample form into the appropriate tables. [4 marks]

```
INSERT INTO StudentInfo (StudentID, Name, Address, Email, Degree)
VALUES(
        12345678,
        ‘James Bond’,
        ‘10 Downing Street, ‘Wellington’,
        ‘jamesbond@gmail.com’,
        ‘Bachelor of Computer Science’
 );

INSERT INTO CourseInfo (Course, CourseName)
VALUES
        ( ‘INFO101’, ‘Fundamentals of IS’ ),
        ( ‘FCOM100’, ‘NZ Business Enviro’ ),
        ( ‘ELCM203’, ‘Internet Apps’ ),
        ( ‘ELCM222’, ‘E-business Strategy’ );

INSERT INTO GradeInfo (StudentID, Course, Year, Grade)
VALUES
        ( 12345678, ‘INFO101’, 2013, ‘F’ ),
        ( 12345678, ‘FCOM100’, 2013, ‘C’ ),
        ( 12345678, ‘ELCM222’, 2015, ‘B+’ ),
        ( 12345678, ‘INFO101’, 2014, ‘A’ ),
        ( 12345678, ‘ELCM203’, 2013, ‘C’ );
```

h. Write the SQL to find the total number of ‘F’ grade for each student. The output should display Name, Student ID, and the “Total number of F Grade”. Use the table names provided in (f). Use only the “JOIN ON” method for joining tables. [4 marks]

```
SELECT Name, StudentID, COUNT(Grade) AS “Total number of ‘F’ grade for each
          student”
FROM StudentInfo s JOIN CourseInfo c JOIN GradeInfo g ON s.StudentID =
          g.StudentID AND c.Course = g.Course
WHERE Grade = ‘F’
```

i. If the course name is possible to change after each year but not the course code, show the new ‘CourseInfo’ dependency diagram. [2 marks]



3. A table to record the project accounts of employee is given below:

a. Explain clearly with examples why the above table is not a good database solution with respect to insert, delete and update anomalies. [6 marks]

   Update anomaly:
   If we update a project, then many employees will have their project affected. Vice versa for updating employees in relation to projects. This can get confusing to understand because we might lose track of what has been updated and which entries might have modified values. In example, if we update *PROJ_ID 13* to *12*, we will no longer know which *EMP_ID* worked in *PROJ_ID 13* as all *EMP_ID* of *PROJ_ID 13* have been assigned to *PROJ_ID 12,* which is bad.

   Delete anomaly:
   If we delete a project, then many employees will lose their relationship. Vice versa for deleting employees in relation to projects. This can get confusing to understand because we might lose track of what has been deleted. In example, if we delete *PROJ_ID 13* and say *EMP_ID* 3 was composite with *PROJ_ID 13*, we will also lose all information about *EMP_ID 3*, which is bad.

   Insert anomaly:
   If we have a fresh new employee, they may have no projects assigned to them. This means we would need to create a dummy *PROJ_ID* value in order to add this new *EMP_ID* value into this table. Then we would have to replace the *PROJ_ID* of the *EMP_ID* with the real value when the employee is finally assigned a project. That is, if the database master remembers that there is a dummy *PROJ_ID* value, which is bad.

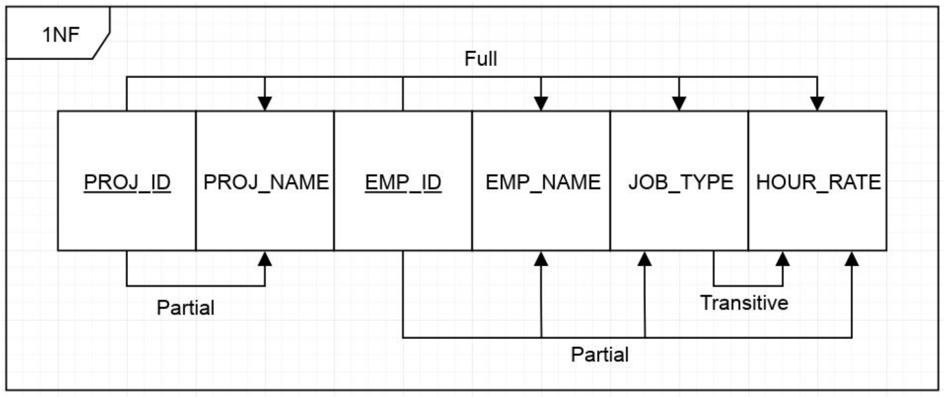
b. Identity the primary key for the above table (use this type of representation, Primary Key (A, B)). [2 marks]

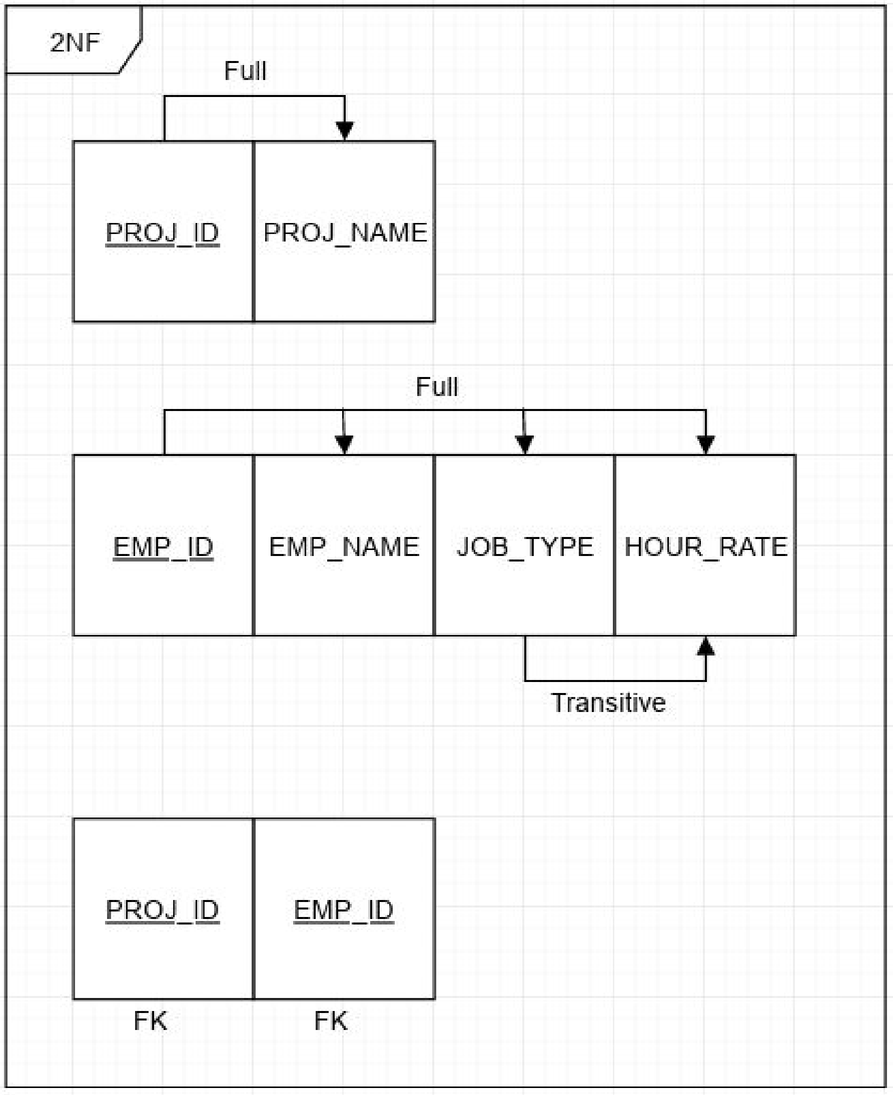
   Primary Key (EMP_ID, PROJ_ID)

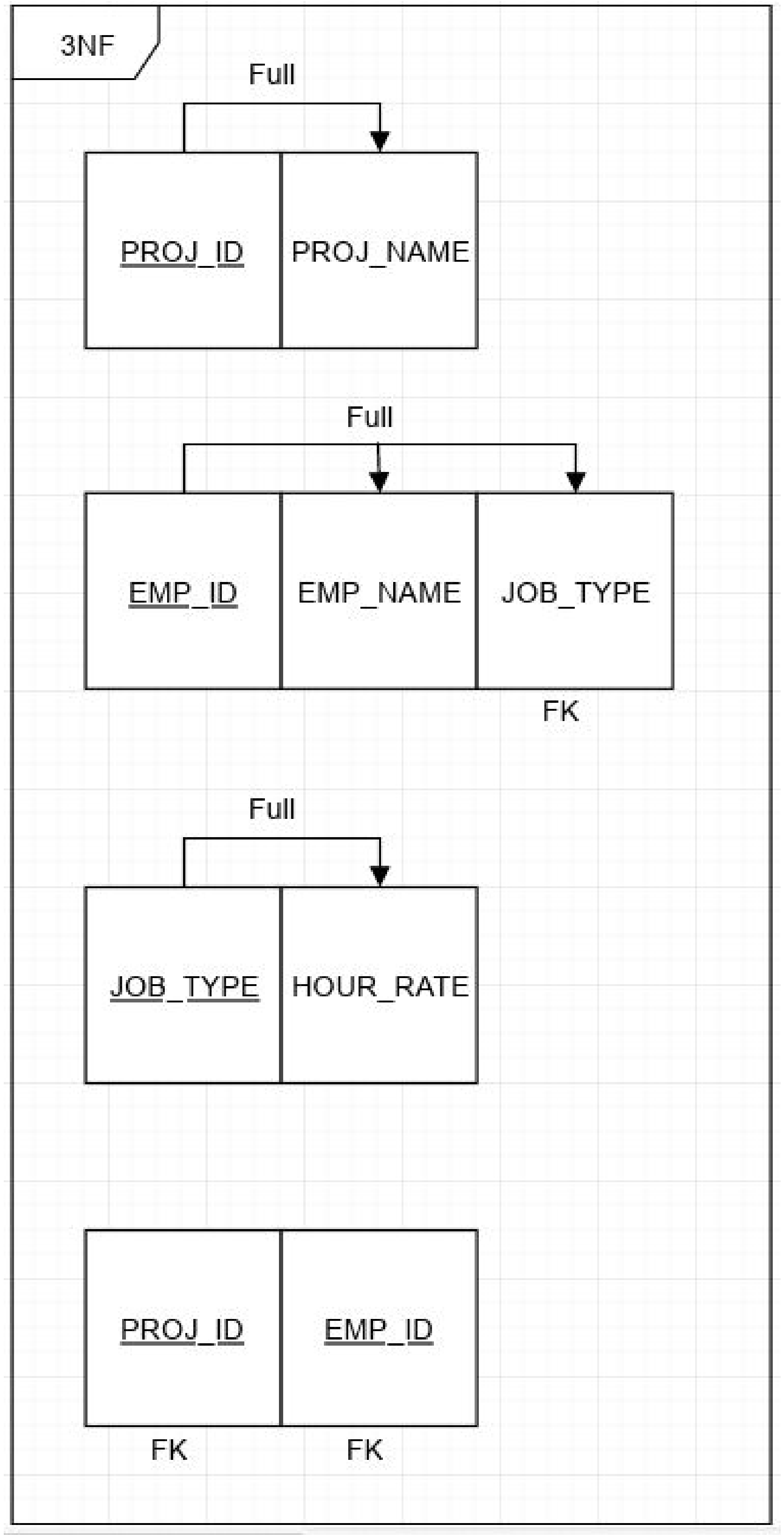
c. Identify the Transitive dependency in the above table (use this type of representation, A→B). [2 marks]

   JOB_TYPE→HOUR_RATE

d. Provide a normalised 3NF solution. You must show the progress from 1NF to 2NF, and then to 3NF. Indicate all the primary keys and foreign keys in the normalised tables. [10 marks]






3NF
Full
PROJ_ID
PROJ_NAME
Full
EMP_ID
EMP_NAME
JOB_TYPE
FK
Full
JOB_TYPE
HOUR_RATE
PROJ_ID
EMP_ID
FK
FK