BUAN 6320 Database Foundations **Amritha Rekha**
Assignment #1 – Intro SQL **AXR180076**

Chapter 1 Problems – Database Systems

Do Problems 1-8 on page 32 of our textbook using the data provided in the book figures.

Answer:

1. The file structure given in the figure contains 7 records. There are 5 fields per record present in the file structure.

2. If we want to produce the listing by city, it would be a cumbersome process because the city name is included inside the field MANAGER_ADDRESS and it is difficult to extract the city name for each of the 7 records as per there index. To fix this problem, we need to alter the file structure by splitting the MANGER_ADDRESS field into ADDRESS_LINE1, ADDRESS_LINE2, CITY, STATE, and ZIPCODE so that it would be easy to produce a listing by city using the CITY field

3. To produce a listing of file contents by last name , area code, city, state, or zip code, we need to split three existing fields in the file structure i.e. the PROJECT_MANAGER, MANAGER_PHONE and MANAGER_ADDRESS.

   The PROJECT_MANAGER field needs to be splitted into three fields such as PROJECT_MANAGER_FNAME, PROJECT_MANAGER_MNAME and PROJECT_MANAGER_LNAME. So that we would list the file by last name.
   The MANAGER_ADDRESS field needs to splitted into six fields ADDRESS_LINE1, ADDRESS_LINE2, CITY, STATE, and ZIPCODE, so that we could use the revelant fields like CITY, STATE and ZIPCODE to do the listing are per required.
   The MANGER_PHONE field also needs to splitted into 2 fields llike AREA_ CODE & PHONE_NO separately. With the above-mentioned alterations, we could produce the listing as needed.

4. In the file structure provided data redundancy occurs for 2 records. We could see that the PROJECT_MANAGER = Holly B.Parker appears three times in the file structure with same data entries for MANAGER_PHONE and MANAGER_ADDRESS and only the PROJECT_CODE and PROJECT_BID_PRICE varies that means the personal information of the manager is the same , but Holly B.Parker is assigned with three projects: 21-5Z, 25-9T and 29-2D.

We could also detect data redundancy appeared for PROJECT_MANAGER = George F.Dorts appears two times in the file structure with the same data entries in MANAGER_PHONE and MANAGER_ADDRESS and here again the PROJECT_CODE and PROJECT_BID_PRICE varies indicating that it is not that the personal information that repeats but the manager is associated with two projects : 25-5A and 27-4Q.

These redundancies could lead to different update,deletion or insertion anomalies. For example, if there is an update in George F.Dorts address or phone number ,it must be changed in two rows and if the system skips to update it in one of the rows this would lead to updation anomaly. So, it is always better to keep redundant-free data.

5. The data redundancy in the file structure appears in multiple areas.First of all, when we are considering the employees and their allocation to the project, here some of the employees like John D.Newson, David F.Schwaan and Anne R. Raamoras is allocated to two projects each. These would lead to various types to anomalies like if need to update the phone no of EMP_NUM =101 , we need to update it in two places. So, if we miss the update in one area this would lead to anomalies. Secondly instead of mentioning the project names multiple times we could maintain a separate file structute for projects having the details like PROJ_NUM,PROJ_NAME so that we need to mention only either of them in this file structure and would also save the data in some scenarios i.e. if EMP_NUM = 101 and 108 left the company due to some reason then the whole record gets deleted that means the existence of the PROJ_NAME= Coast gets wiped off from the file structure. Thirdly we could see that the job charge per hour differs for the JOB_CODE = CT, it appears as three different values 60, 62, 26. Here we don't know whether this change in value is due to any job position / category difference. Hence the file structure lacks a proper clarity and management. So, to prevent these data redundancies, separate file structure should be mainained like an Employee file to store EMP_NUM , EMP_NAME & EMP_PHONE, a Project file to store the PROJ_NAME & PROJ_NUM and a Job file structure to record the job details so that the original data would retain until unless it is removed completely.

6. I would recommend splitting up the EMP_NAME field into three fields to store the first, middle and last name separately in EMP_FNAME,EMP_MNAME and EMP_LNAME respectively. Similarily we could also split the EMP_PHONE into AREA_CODE and NUMBER seperately.

7. Project data, Employee data and the Job data are the different data sources present in the file structure. The project data consists of the PROJ_NUM, PROJ_NAME,PROJ_HOURS. The employee data comprises of EMP_NUM, EMP_NAME and EMP_PHONE. The job data conists of the JOB_CODE and JOB_CHG_HOUR.

8. Instead of maintining all these data in one single file structure to eliminate redundancy we must maintain separate files based on the data it stores. In this example we must

divide the single file structure into Emplyee, Project and Job file structures. The Employee file should contain EMP_NUM, EMP_NAME and EMP_PHONE.
The Job file must contain JOB_CODE and JOB_CHG_HOUR. The project file consists of the PROJ_NUM, PROJ_NAME and PROJ_HOURS

Chapter 2 Problems – Database Models

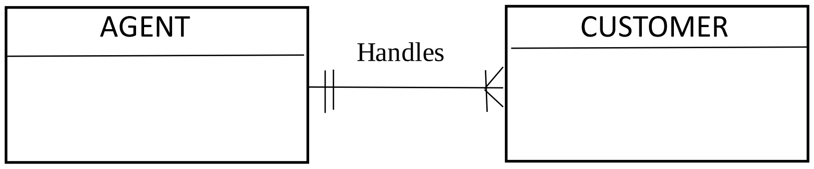
Do Problems 1-9 on pages 64-65 of our textbook using the data provided in the book figures.

Answer:

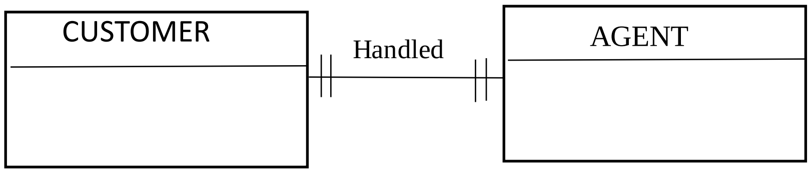
1. Business rule 1 : " An AGENT handles multiple CUSTOMERS" (1 : M)
   Business rule 2 : "A CUSTOMER is handled by one AGENT " (1 : 1)

2. **Crow-foot Entity relation diagram**

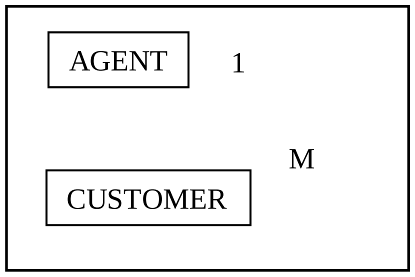
   Business rule 1 – ER relationship is One to Many



   Business rule 2 – ER relationship is One to One

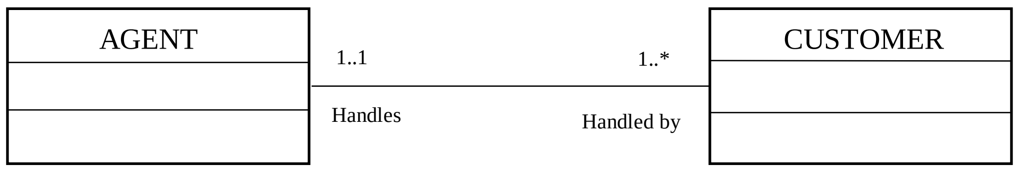


3. **Object Represesntation**

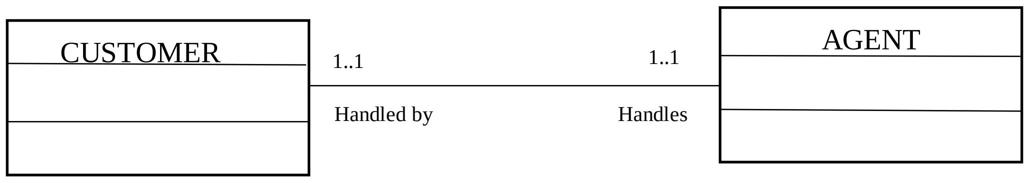


   UML class diagram

   Business rule 1 – UML class diagram representing One to Many



Business rule 2 – UML class diagram representing One to One

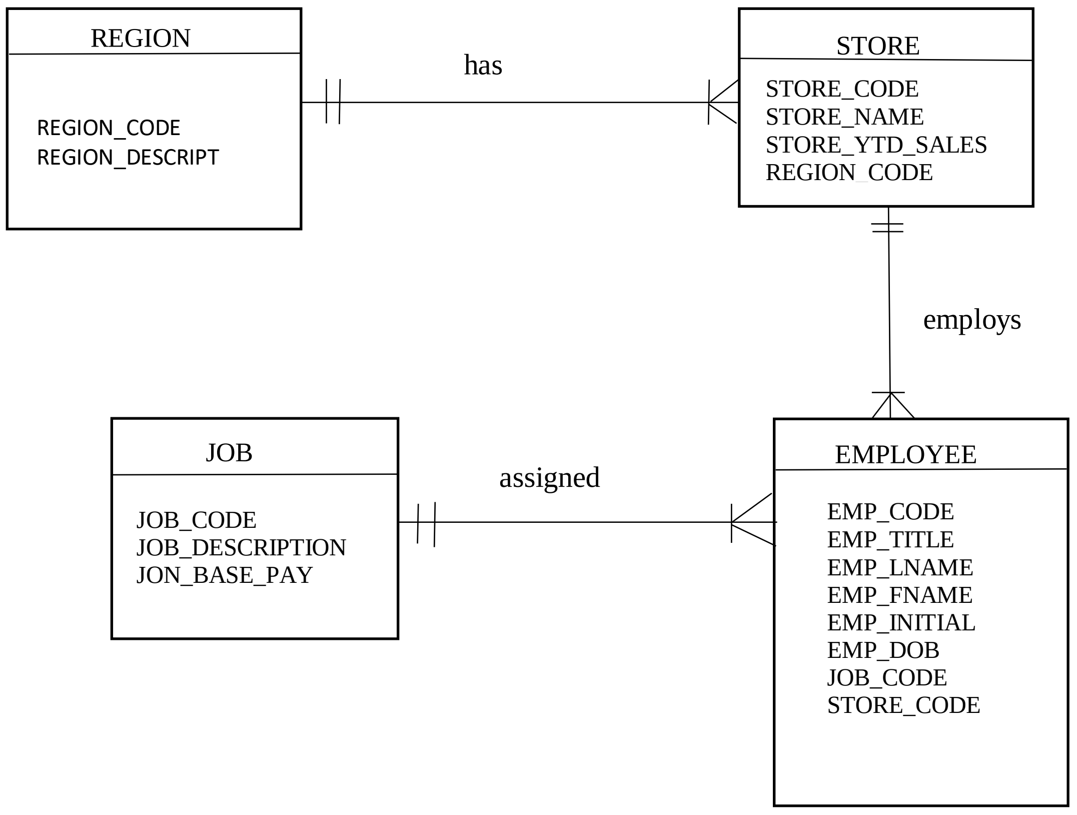


4. Business Rule 1 : One REGION can *have* many STORES
   Relationship – One to Many (1:M)

   Business Rule 2 : One STORE *employs* many EMPLOYEES
   Relationship – One to Many (1:M)

   Business Rule 3 : Many EMPLOYEES can be *assigned* the same JOB function
   Relationship – Many to One (M:1)

5. Crow Foot Entity Relationship diagram for DealCo

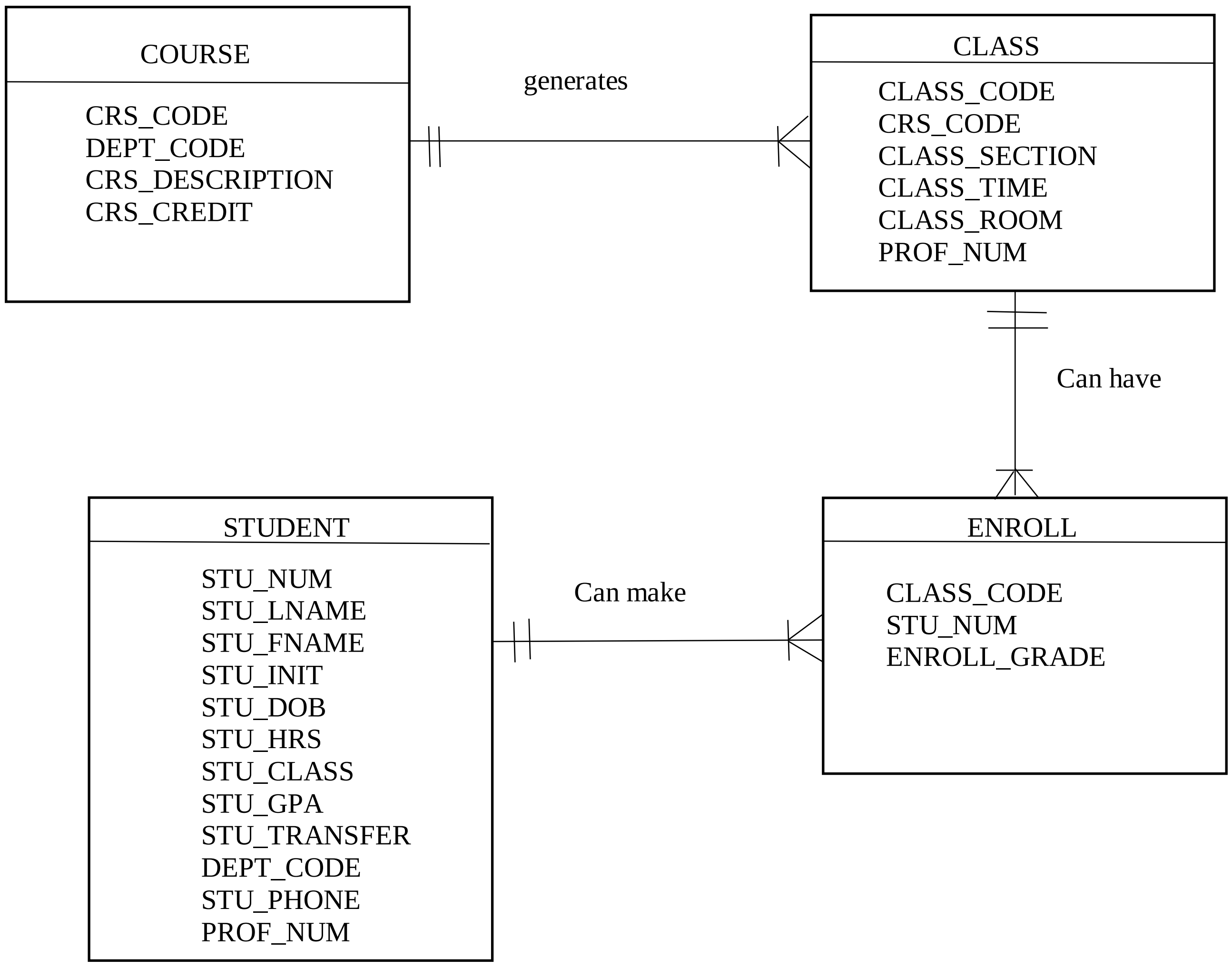


6. Business Rule 1 : One COURSE *generates* many CLASSES
   Relationship – One to Many (1:M). For example, the course Database foundations by James scott is having multiple class schedules on Tuesday,Thursday and Saturday

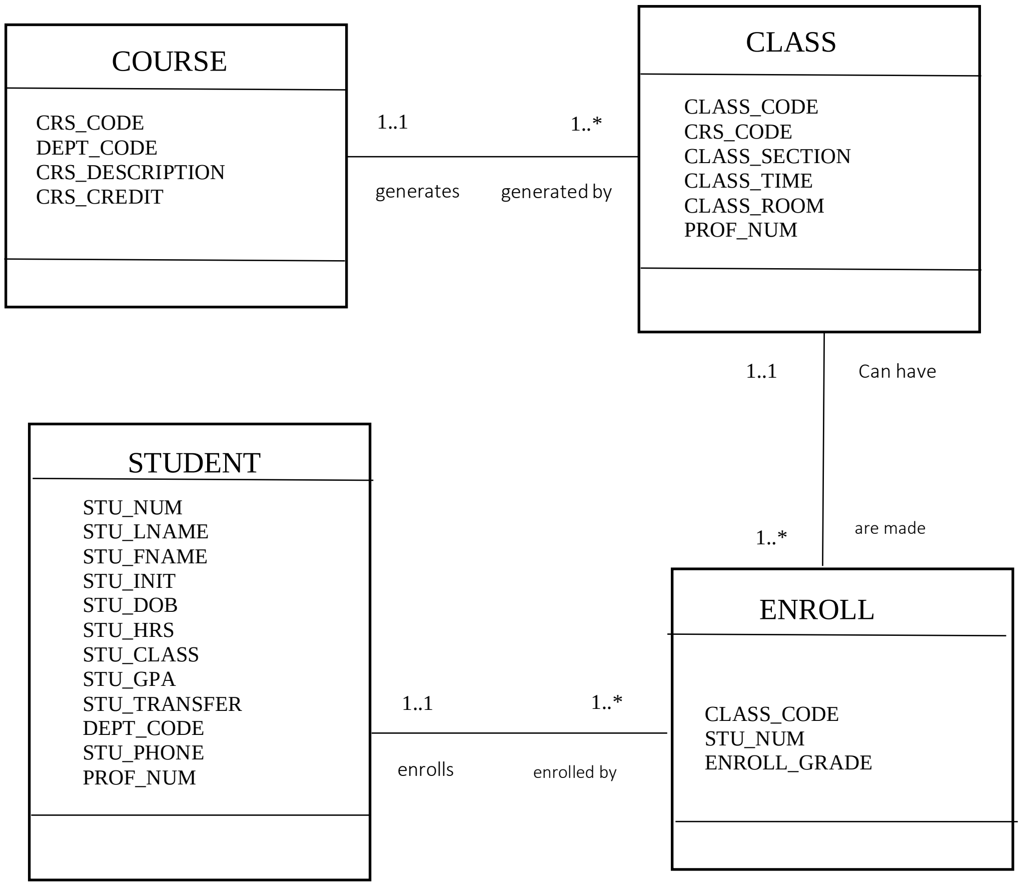
   Business Rule 2 : One CLASS can *mention* multiple ENROLLMENTS.
   Relationship – One to Many (1:M). For instance, Student A and Student B can enroll to the same class.

   Business Rule 3 : One STUDENT can *make* multiple ENROLLMENTS for different subjects.
   Relationship – One to Many (1:M). For example, Student A can enroll to Database foundations as well asBusiness analytics with R.

7. Crow Foot Entity Relationship diagram for Tiny College



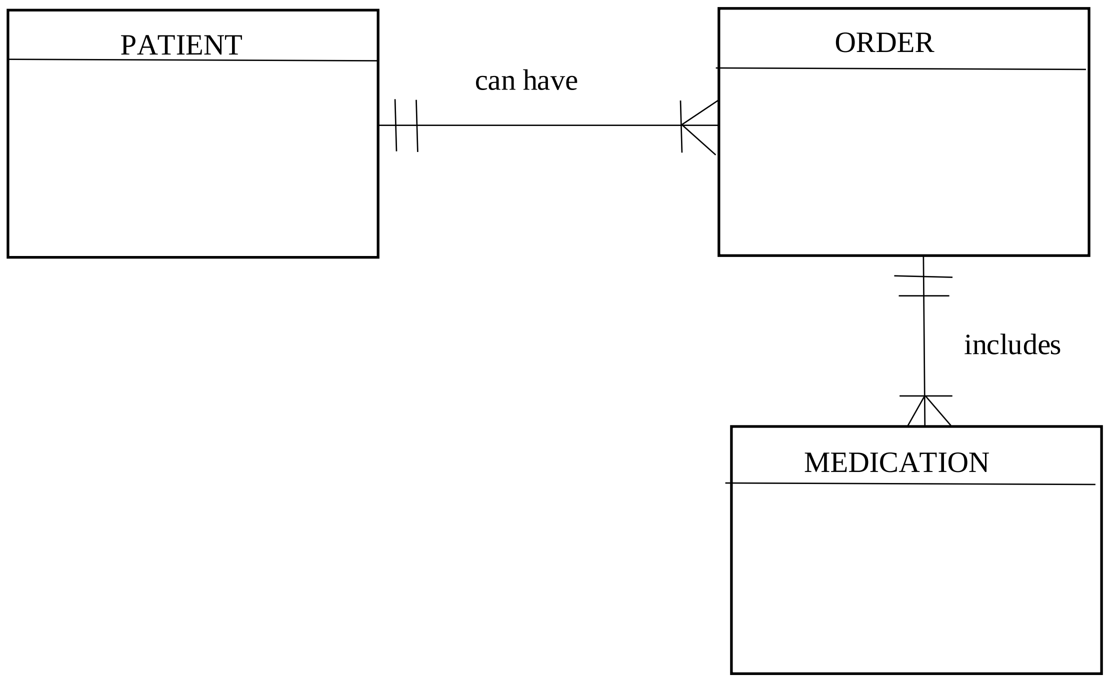
8. UML Class Diagram



9. a) Business Rule 1 : A PATIENT *can have* multiple ORDERS

Business Rule 2 : An ORDER *includes* several MEDICATIONS

b) Crow foot ERD diagram



For the following SQL coding problems use the UniversityDDL Database provided by instructor.

Problem #1 – Retrieving all rows and all columns from a table

For each of our tables, retrieve all rows and all columns.
Tables are Student, Faculty, Offering, Course, and Enrollment
(no need to sort at this point)

Answer:
SELECT * FROM Student S;
SELECT * FROM Faculty F;
SELECT * FROM Offering O;
SELECT * FROM Course C;
SELECT * FROM Enrollment E;

Problem #2 – Retrieving a subset of columns from a table and sorting them both with and without the ASC keyword

Retrieve the student number, student first name, and student last name for all students

Sort the results by student last name then by student first name
Use the ASC keyword on the query

Answer:
SELECT S.STDNO, S.STDFIRSTNAME, S.STDLASTNAME FROM Student S
ORDER BY S.STDLASTNAME ASC, S.STDFIRSTNAME ASC;

Repeat the query omitting ASC

Answer:
SELECT S.STDNO,S.STDFIRSTNAME,S.STDLASTNAME FROM Student S
ORDER BY S.STDLASTNAME, S.STDFIRSTNAME;

Problem #3 – Retrieving a subset of columns from a table and sorting them on multiple columns mixing ascending and descending order, using both named and positional notation

Retrieve the student last name, student first name, and GPA for all students
Sort the results by GPA highest first, then by student last name, then by student first name
Use column names to sort (omit ASC)

Answer:
SELECT S.STDLASTNAME, S.STDFIRSTNAME, S.STDGPA FROM Student S
ORDER BY S.STDGPA DESC, S.STDLASTNAME, S.STDFIRSTNAME;

Repeat the query using positional notation

Answer:
SELECT S.STDLASTNAME, S.STDFIRSTNAME, S.STDGPA FROM Student S
ORDER BY 3 DESC, 1, 2;

Problem #4 – Retrieving columns from a table both with and without duplicates

Retrieve the student city and class for all students with duplicates
Repeat query without duplicates

Answer:
SELECT S.STDCITY, S.STDCLASS FROM Student S;

Repeat query without duplicates

Answer:
SELECT DISTINCT S.STDCITY, S.STDCLASS FROM Student S;

Problem #5 – Retrieving a subset of rows with a single Boolean expression

Retrieve the student last name, student first name, and GPA for all students with a GPA greater than 3.2

Answer:
SELECT S.STDLASTNAME, S.STDFIRSTNAME, S.STDGPA FROM Student S WHERE S.STDGPA > 3.2

Problem #6 – Retrieving a subset of rows with multiple complex Boolean expressions

Retrieve the student last name, student first name, and GPA for all students with a GPA (more than 2.2 and less than 2.7) OR (more than 3.2 and less than 3.8)

Answer:
```
SELECT S.STDLASTNAME, S.STDFIRSTNAME, S.STDGPA
FROM Student S
WHERE ((S.STDGPA > 2.2 AND S.STDGPA < 2.7) OR (S.STDGPA > 3.2 AND S.STDGPA < 3.8));
```

Problem #7 – Retrieving a subset of rows with the BETWEEN operator

Retrieve the student last name, student first name, and GPA for all students with a GPA that is between 2.7 and 3.2 inclusive

Answer:
```
SELECT S.STDLASTNAME, S.STDFIRSTNAME, S.STDGPA
FROM Student S
WHERE S.STDGPA BETWEEN 2.7 AND 3.2;
```

Problem #8 – Retrieving a subset of rows with testing for NULLs

Retrieve the offer number, course number, year, and faculty number from all course offerings that has not yet been assigned a Faculty

Answer:
```
SELECT O.OFFERNO, O.COURSENO, O.OFFYEAR, O.FACNO
FROM Offering O
WHERE O.FACNO IS NULL;
```

Repeat query for course offerings that have been assigned a Faculty

Answer:

```
SELECT O.OFFERNO, O.COURSENO, O.OFFYEAR, O. FACNO
FROM Offering O
WHERE O.FACNO IS NOT NULL;
```