Matthew Maya 11/02/2021

Part 1

A. **Main Entity Types:** Department, Student, Major, and Event

B. **Main Relationship:**

Student ------attends-----> Event,
Student ------is studying------>Major
Major --------belongs to-------->Department
Department -------hosts------>Event

C. **Multiplicity Constraints:**

Student (1....*)------attends-----> Event(1…*)
Student (1…*)------is studying-----><u>Major</u> (1…*)
Major (1…*)--------belongs to-------><u>Department</u>(1…1)
Department(1...*)-------hosts------>Event(0...*)
Department(1...*) ---------Has/Has a -------> Student(0...*)

D. **Attributes:**

Student: Major(s), name, and initials
- **Student**(studentID, firstName, lastName, initials)

Major: Major Name(s), Department it is associated with, Major code
- **Major**(majorID, majorName, majorCode)

Department: Department Name, Chair Name, # of Faculty per department
- **Department**(deptName, chairName, noFacultyPerDept)

Event: Event Name, Start Date, End Date
- **Event**(eventID, eventName, startDate, endDate)

E. **Candidate & Primary Key:**

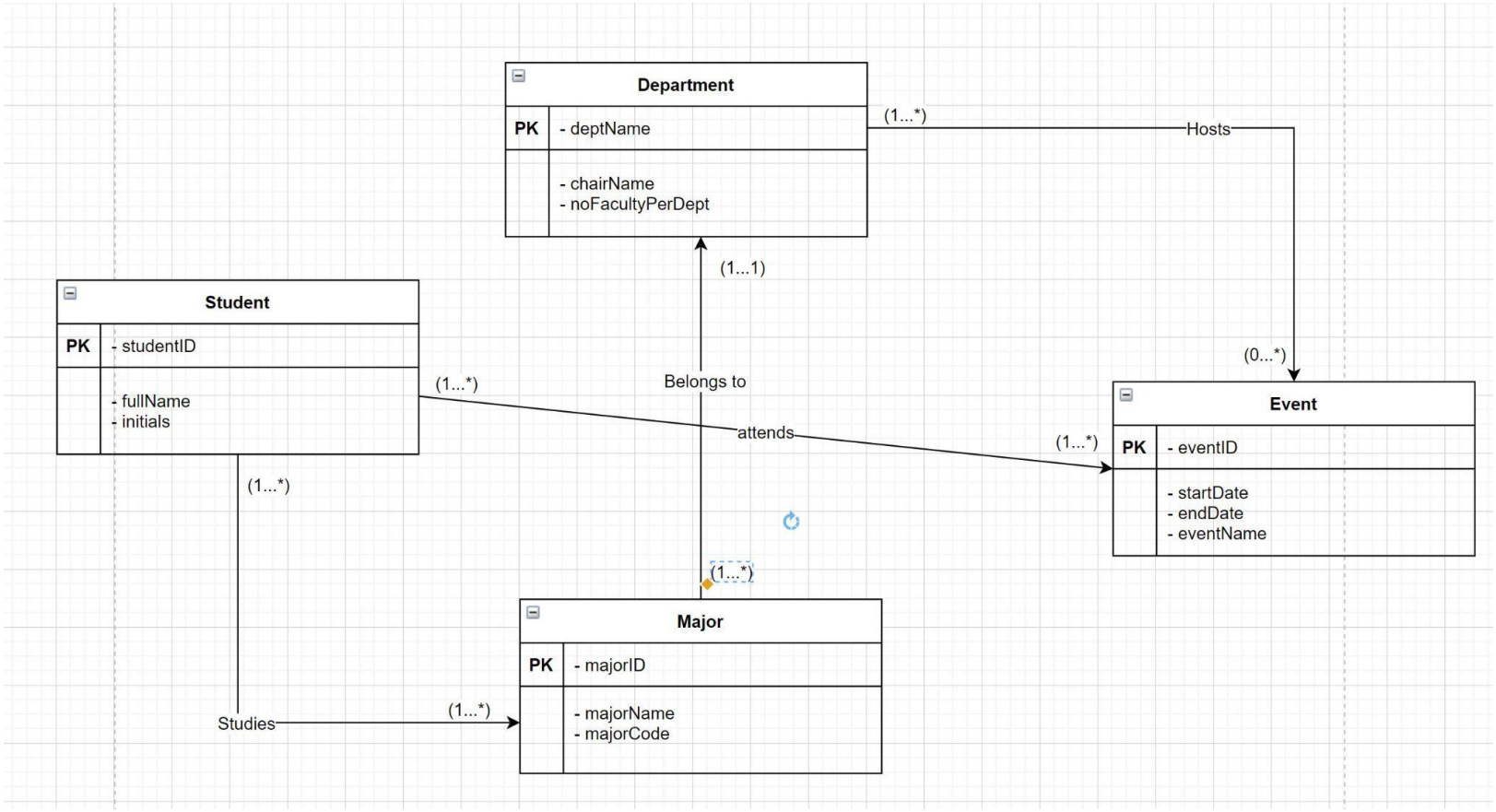
Department: Department Name, Chair Name, # of Faculty per department
Candidate Key(s): deptName
Primary Key: deptName
*There can't be two departments with the same name at a university.

Event: Event ID, Event Name, Start Date, End Date
Primary Key: eventID
*No two students can have the same event ID.

Major: Major ID, Major Name(s), Department it is associated with, Major code
Candidate Key(s): majorCode
* There are no two major codes which are the same. Major Code is unique.
Primary Key: majorID
*No two students can have the same major ID.

Student: Student ID, Major(s), firstName, lastName, and initials
Primary Key: StudentID
*No two students can have the same Student ID.

F.



Part 2

A.

i. There is a *:* relationship between the **Student** table and the **Major** table where the primary keys (majorID, studentID) from both tables are added in the new relation labeled as **academics**.

- Academics(**studentID**, **majorID**)

ii. There is a *:* relationship between the **Event** table and the **Student** table where the primary keys (studentID, eventID) from both tables are added in the new relation labeled as **attendance**.

- Attendance(**studentID**, **eventID**)

iii. There is a 1:* relationship between the **Major** table (Child Entity) and the **Department** table (Parent Entity) where the primary key (deptName) in the Department table is the foreign key in the Major table.

iv. There is a *:* relationship between the **Department** table and **Event** table where the primary keys (deptName, eventID) from both tables are added in the new relation labeled as academics.
    - Hostee(**deptName**, **eventID**)

v. **Student**(studentID, firstName, lastName, initials)

vi. **Major**(majorID, majorName, majorCode)

vii. **Event**(eventID, eventName, startDate, endDate)

viii. **Department**(deptName, chairName, noFacultyPerDept)

B. Functional Dependencies in Student

studentID → firstName, lastName, initials (Primary Key)

firstName, lastName → initials (Transitive Dependency)

A very particular situation in which a person's name would be unique in the table (because it does not represent a specific person but everyone named "X") in which we would store the name and the initials. This adds up a lot of complexity compared to the redundancy it removes (the initials column). It is best to keep it in the student table because of the extra calculations joining the two tables would imply.

Functional Dependencies in Major

majorID → majorName, majorCode (Primary Key)

majorCode → MajorName, majorID (Candidate Key)

Functional Dependencies in Department

deptName → chairName, noFacultyPerDept (Primary Key)

Functional Dependencies in Event

eventID → startDate, endDate, eventName (Primary Key)

Functional Dependencies in Academics

N/A

Functional Dependencies in Attendance

N/A

Functional Dependencies in Hostee

N/A

In order for the ER diagram to be in third normal form the diagram must follow all the rules of a third normal form as well as all the rules for the previous normal forms. The diagram follows the rules to qualify it as third normal form as it has eliminated all the transitive dependencies aside from fullName → initials due to complexity. It also follows the rules for second normal form by eliminating all the partial dependencies. Lastly the diagram contains one and only one value per intersection of row and column meaning it also follows the rules of first normal form. Following all these rules qualifies this diagram as being of third normal form.

C. Validate the logical model against user transactions

List all students and their names that study Biology.

The first step to finding all students and their names who study Biology is to triple join the Major, Academics, and Student tables. Then you would list the studentID, fName, lName where Academics majorID = Major majorID, Academics studentID = Student studentID, and majorName ~= 'Biology' (or the code = 'BIO')

List the names of the department(s) that hosted the Walk to Defeat ALS event in 2021.

The first step to list the names of the department(s) that hosted the Walk to Defeat ALS event in 2021 is to join the Department, Hostee, and Event tables. Next you would list deptName where eventName = 'Walk to Defeat ALS' & startDate > '12-31-2020'.

List all the information relating to the majors under the Arts & Science department.

First the user would have to join the Department and Major table. Next the user would return the majorName, majorCode, majorID where Department deptName = Major deptName and deptName = 'Arts & Science'.

Find the number of faculty in the Engineering Department

In order to do this the user would have to join the Department and Major tables. Next they would return noFacultyPerDepartment where Department deptName = Major deptName and deptName = 'Engineering'.

List all students and their names who attended a Black Lives Matter event.

First you would have to triple join the Student, Attendance, and Event tables. Next you would return the studentID, fName, lName where Attendance studentID = Student studentID, Attendance eventID = Event eventID, and where the Event eventName ~= "Black Lives Matter".

D. Integrity Constraints
   i. Primary key Constraints: None of the primary keys can be null and each primary key must be unique.
   ii. Referential integrity/Foreign key constraints are listed below under iii.)
   iii. **Student**(studentID, firstName, lastName, initials)
   Primary Key studentID

Department(deptName, chairName, noFacultyPerDept)
Primary Key deptName

Event(eventID, eventName, startDate, endDate)
Primary Key eventID

Major(majorID, majorName, majorCode)
Primary key majorID
Alternate key majorCode
Foreign Key deptName **references** Department(deptName) ON UPDATE CASCADE ON DELETE NO ACTION

Academics(studentID, majorID)
Primary key studentID, majorID
Foreign Key studentID **references** Student(studentID) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key majorID **references** Major(majorID) ON UPDATE CASCADE ON DELETE NO ACTION

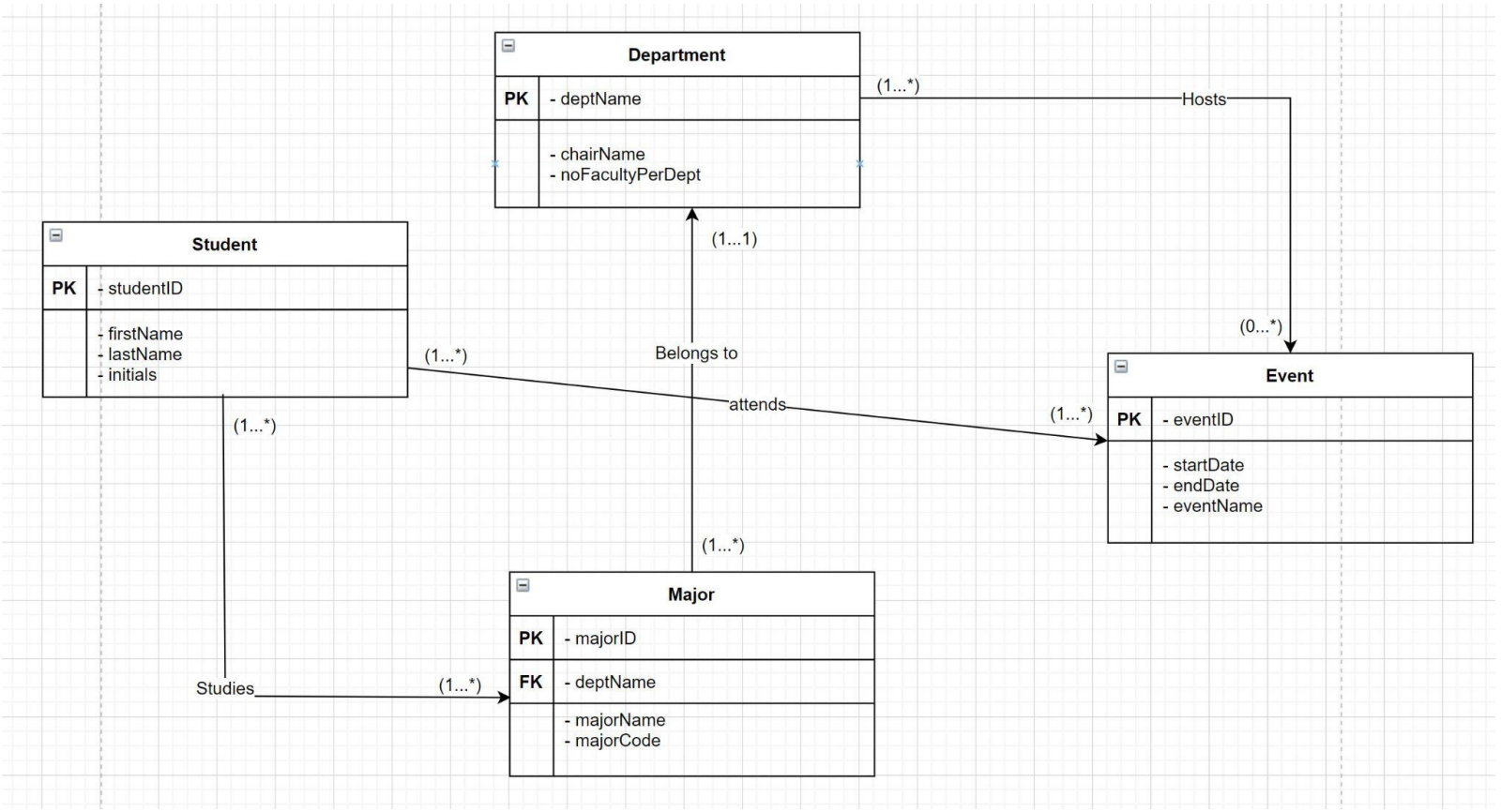
Attendance(studentID, eventID)
Primary key studentID, eventID
Foreign Key studentID **references** Student(studentID) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key eventID **references** Event(eventID) ON UPDATE CASCADE ON DELETE NO ACTION

Hostee(deptName, eventID)
Primary key deptName, eventID
Foreign Key deptName **references** Depart(deptName) ON UPDATE CASCADE ON DELETE NO ACTION
Foreign Key eventID **references** Event(eventID) ON UPDATE CASCADE ON DELETE NO ACTION

iv. General Constraints:

- chairName can't be NULL in Department
- Initials has to be greater than one character
- startDate must be after a past end time or current date
- endDate has to be greater than the startDate
- majorCode has to be three characters

E. ER Diagram



Part 3

A.)

```
CREATE TABLE Department
(deptName VARCHAR(100),
chairName VARCHAR(50) NOT NULL,
noFacultyPerDept INT,
PRIMARY KEY (deptName),
CONSTRAINT deptNameSyntax
   Check(deptName LIKE 'Department%'));

CREATE TABLE Student
(studentID VARCHAR(100),
firstName VARCHAR(100) NOT NULL,
lastName VARCHAR(100) NOT NULL,
initials VARCHAR(3),
PRIMARY KEY (studentID),
CONSTRAINT initialsLength
   Check(LENGTH(initials) > 1 ));
```

```
CREATE TABLE Major
(majorID VARCHAR(15),
majorName VARCHAR(100) NOT NULL,
majorCode VARCHAR(3) UNIQUE NOT NULL,
deptName VARCHAR(100) NOT NULL,
CONSTRAINT majorCodeRequiredLength
   Check(LENGTH(majorCode) = 3 ),
PRIMARY KEY (majorID),
Foreign Key (deptName) references Department ON DELETE CASCADE);

CREATE TABLE Event
(eventID VARCHAR(12),
eventName VARCHAR(100) NOT NULL,
startDate DATE,
endDate DATE,
CONSTRAINT endDateGreater
   Check(endDate > startDate),
CONSTRAINT startDateGreater
   Check(startDate > '11-DEC-2021'),
PRIMARY KEY (eventID));

CREATE TABLE Academics
(studentID VARCHAR(100),
majorID VARCHAR(15),
PRIMARY KEY (studentID, majorID),
Foreign Key (studentID) references Student ON DELETE CASCADE,
Foreign Key (majorID) references Major ON DELETE CASCADE);

CREATE TABLE Attendance
(studentID VARCHAR(100),
eventID VARCHAR(12),
PRIMARY KEY(studentID, eventID),
FOREIGN KEY(studentID) REFERENCES Student ON DELETE CASCADE,
FOREIGN KEY(eventID) REFERENCES Event ON DELETE CASCADE);

CREATE TABLE Hostee
(deptName VARCHAR(100),
eventID VARCHAR(12),
PRIMARY KEY(deptName, eventID),
FOREIGN KEY(deptName) REFERENCES Department ON DELETE CASCADE,
FOREIGN KEY(eventID) REFERENCES Event ON DELETE CASCADE);
```

B.)

```
INSERT INTO Student
```

Values('S001', 'Matthew', 'Maya', 'MM');

INSERT INTO Student

Values('S002', 'Nicky', 'Sosnivka', 'NS');

INSERT INTO Student

Values('S003', 'Gabe', 'Simmons', 'GB');

INSERT INTO Student

Values('S004', 'Paul', 'Rhoades', 'PR');

INSERT INTO Student

Values('S005', 'Juan', 'Arango', 'JA');

STUDENTID	FIRSTNAME	LASTNAME	INITIALS
S001	Matthew	Maya	MM
S002	Nicky	Sosnivka	NS
S003	Gabe	Simmons	GB
S004	Paul	Rhoades	PR
S005	Juan	Arango	JA

INSERT INTO Department

VALUES ('Department of Arts & Science', 'Steve Jobs', 92);

INSERT INTO Department

VALUES ('Department of Communications', 'Oprah Winfrey', 103);

INSERT INTO Department

VALUES ('Department of Engineering', 'Elon Musk', 34);

INSERT INTO Department

VALUES ('Department of Marine & Atmospheric Science', 'Steve Irwin', 57);

INSERT INTO Department

VALUES ('Department of Business', 'Jordan Belfort', 65);

DEPTNAME	CHAIRNAME	NOFACULTYPERDEPT
Department of Arts & Science	Steve Jobs	92
Department of Communications	Oprah Winfrey	103
Department of Engineering	Elon Musk	34
Department of Marine & Atmospheric Science	Steve Irwin	57
Department of Business	Jordan Belfort	65

INSERT INTO Event

VALUES('E001', 'Homecoming', '01-NOV-2022', '07-NOV-2022');

INSERT INTO Event

VALUES('E002', 'Walk to Defeat ALS', '23-MAY-2022', '24-MAY-2022');

INSERT INTO Event

VALUES('E003', 'Welcome Week Festival', '26-AUG-2022', '27-AUG-2022');

INSERT INTO Event

VALUES('E004', 'Black Lives Matter', '01-FEB-2022', '01-MAR-2022');

INSERT INTO Event

VALUES('E005', 'Walk to End Poverty', '15-JAN-2022', '16-JAN-2022');

EVENTID	EVENTNAME	STARTDATE	ENDDATE
E001	Homecoming	01-NOV-22	07-NOV-22
E002	Walk to Defeat ALS	23-MAY-22	24-MAY-22
E003	Welcome Week Festival	26-AUG-22	27-AUG-22
E004	Black Lives Matter	01-FEB-22	01-MAR-22
E005	Walk to End Poverty	15-JAN-22	16-JAN-22

INSERT INTO Major

VALUES('M002', 'Computer Science', 'CSC', 'Department of Engineering');

INSERT INTO Major

VALUES('M003', 'Biology', 'BIO', 'Department of Arts & Science');

INSERT INTO Major

VALUES('M015', 'Marine Biology', 'MAR', 'Department of Marine & Atmospheric Science');

INSERT INTO Major

VALUES('M023', 'Journalism', 'JOU', 'Department of Communications');

INSERT INTO Major

VALUES('M031', 'Marketing', 'MKT', 'Department of Business');

MAJORID	MAJORNAME	MAJORCODE	DEPTNAME
M015	Marine Biology	MAR	Department of Marine & Atmospheric Science
M023	Journalism	JOU	Department of Communications
M002	Computer Science	CSC	Department of Engineering
M003	Biology	BIO	Department of Arts & Science
M031	Marketing	MKT	Department of Business

INSERT INTO Academics

VALUES ('S004', 'M002');

INSERT INTO Academics

VALUES ('S002','M003');

INSERT INTO Academics

VALUES ('S003', 'M015');

INSERT INTO Academics

VALUES ('S001', 'M023');

INSERT INTO Academics

VALUES ('S005', 'M031');

STUDENTID	MAJORID
S001	M023
S002	M003
S003	M015
S004	M002
S005	M031

INSERT INTO Attendance

VALUES ('S002', 'E002');

INSERT INTO Attendance

VALUES ('S004', 'E004');

INSERT INTO Attendance

VALUES ('S001', 'E001');

INSERT INTO Attendance

VALUES ('S003', 'E003');

INSERT INTO Attendance

VALUES ('S005', 'E001');

STUDENTID	EVENTID
S001	E001
S002	E002
S003	E003
S004	E004
S005	E001

INSERT INTO Hostee

VALUES ('Department of Business', 'E001');

INSERT INTO Hostee

VALUES ('Department of Communications', 'E004');

INSERT INTO Hostee

VALUES ('Department of Communications', 'E002');

INSERT INTO Hostee

VALUES ('Department of Marine & Atmospheric Science', 'E004');

INSERT INTO Hostee

VALUES ('Department of Marine & Atmospheric Science', 'E005');

DEPTNAME	EVENTID
Department of Business	E001
Department of Communications	E002
Department of Communications	E004
Department of Marine & Atmospheric Science	E004
Department of Marine & Atmospheric Science	E005

C.)

List all students and their names that study Biology.

SELECT s.studentID, s.firstName, s.lastName
FROM Major m, Academics a, Student s
WHERE a.majorID = m.majorID AND a.studentID = s.studentID AND m.majorCode LIKE 'BIO';

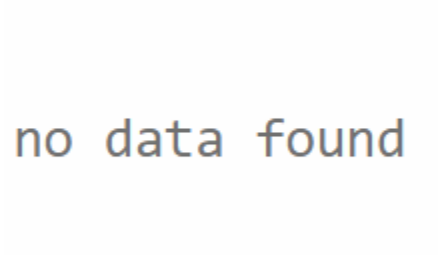
STUDENTID	FIRSTNAME	LASTNAME
S002	Nicky	Sosnivka

List the names of the department(s) that hosted the Walk to Defeat ALS event in 2021.

SELECT h.deptName

FROM Hostee h, Event e

WHERE h.eventID = e.eventID AND e.eventName LIKE 'Walk to Defeat ALS' AND e.startDate > '31-DEC-2020' AND e.endDate < '01-JAN-2022';



List all the information relating to the majors under the Arts & Science department.

SELECT majorName, majorCode, majorID

FROM Department d, Major m

WHERE d.deptName = m.deptName AND d.deptName LIKE '%Arts & Science%';

MAJORNAME	MAJORCODE	MAJORID
Biology	BIO	M003

Find the number of faculty in the Engineering Department

SELECT d.noFacultyPerDept

FROM Department d, Major m

WHERE d.deptName = m.deptName AND d.deptName LIKE 'Department of Engineering';

NOFACULTYPERDEPT
34

List all students and their names who attended a Black Lives Matter event.

SELECT s.studentID, s.firstName, s.lastName

FROM Student s, Attendance a, and Event e

WHERE a.studentID = s.studentID AND a.eventID = e.eventID AND e.eventName LIKE “Black Lives Matter”;

STUDENTID	FIRSTNAME	LASTNAME
S004	Paul	Rhoades

D.) First Two Pictures are of the tables and the final is of the queries

```
  studentID firstName  lastName initials
0      S001   Matthew      Maya       MM
1      S002     Nicky  Sosnivka       NS
2      S003      Gabe   Simmons       GB
3      S004      Paul   Rhoades       PR
4      S005      Juan    Arango       JA
Index(['studentID', 'firstName', 'lastName', 'initials'], dtype='object')

                                  deptName       chairName  noFacultyPerDept
0               Department of Arts & Science      Steve Jobs                92
1               Department of Communications   Oprah Winfrey               103
2                  Department of Engineering       Elon Musk                34
3  Department of Marine & Atmospheric Science     Steve Irwin                57
4                     Department of Business  Jordan Belfort                65
Index(['deptName', 'chairName', 'noFacultyPerDept'], dtype='object')

  eventID              eventName   startDate     endDate
0    E001             Homecoming  2022-11-01  2022-11-07
1    E002     Walk to Defeat ALS  2022-05-23  2022-05-24
2    E003  Welcome Week Festival  2022-08-26  2022-08-27
3    E004     Black Lives Matter  2022-02-01  2022-03-01
4    E005    Walk to End Poverty  2022-01-15  2022-01-16
Index(['eventID', 'eventName', 'startDate', 'endDate'], dtype='object')

  studentID majorID
0      S004    M002
1      S002    M003
2      S003    M015
3      S001    M023
4      S005    M031
Index(['studentID', 'majorID'], dtype='object')
```

```
  studentID majorID
0      S004    M002
1      S002    M003
2      S003    M015
3      S001    M023
4      S005    M031
Index(['studentID', 'majorID'], dtype='object')

  studentID eventID
0      S002    E002
1      S004    E004
2      S001    E001
3      S003    E003
4      S005    E001
Index(['studentID', 'eventID'], dtype='object')

                                  deptName eventID
0                    Department of Business    E001
1               Department of Communications    E004
2               Department of Communications    E002
3  Department of Marine & Atmospheric Science    E004
4  Department of Marine & Atmospheric Science    E005
Index(['deptName', 'eventID'], dtype='object')
```

```
  studentID firstName  lastName
0      S002     Nicky  Sosnivka
Index(['studentID', 'firstName', 'lastName'], dtype='object')

Empty DataFrame
Columns: [deptName]
Index: []
Index(['deptName'], dtype='object')

  majorName majorCode majorID
0   Biology       BIO    M003
Index(['majorName', 'majorCode', 'majorID'], dtype='object')

   noFacultyPerDept
0                34
Index(['noFacultyPerDept'], dtype='object')

  studentID firstName lastName
0      S004      Paul  Rhoades
Index(['studentID', 'firstName', 'lastName'], dtype='object')
```