University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

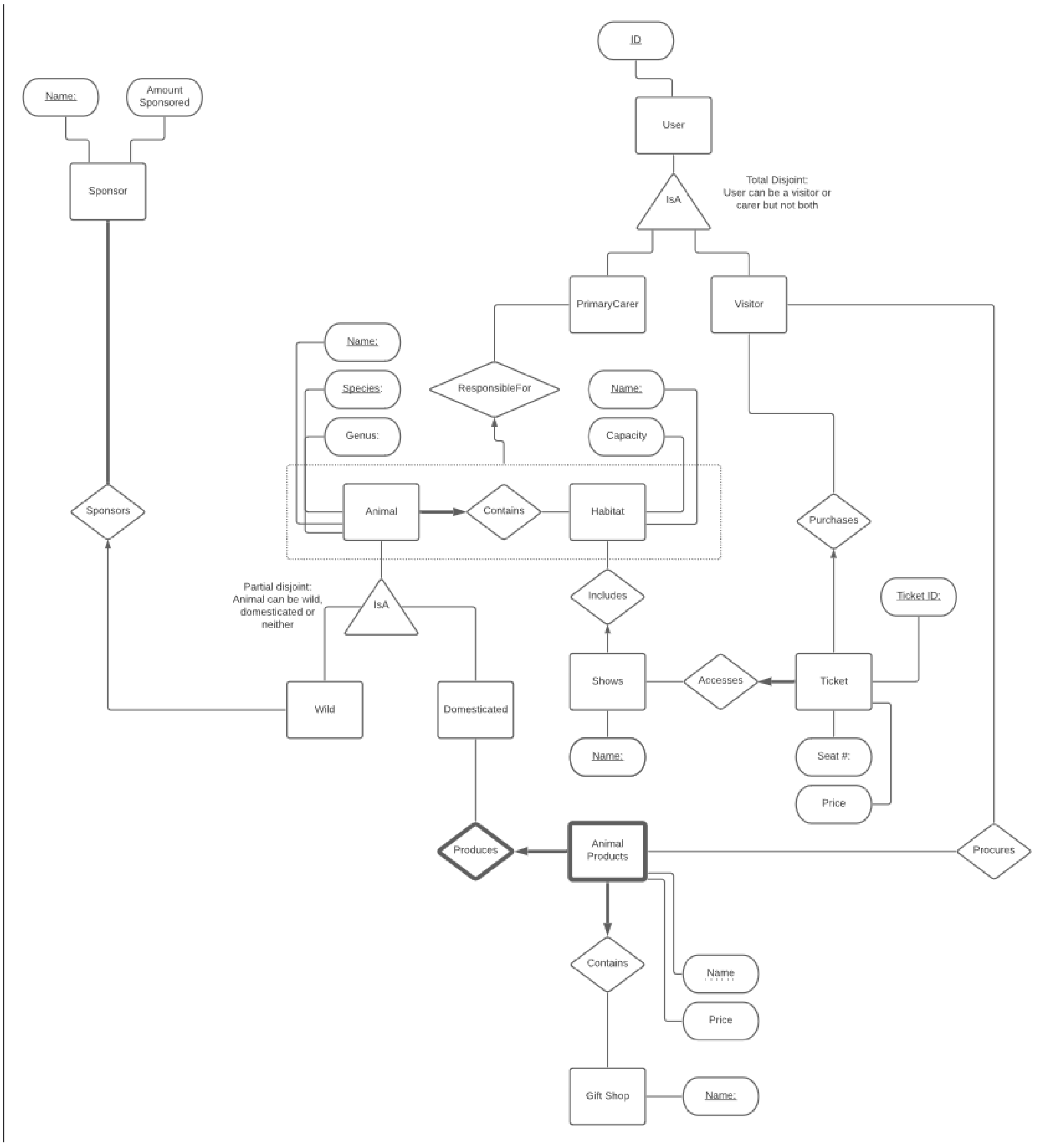
Milestone #: 2

Date: 25 October, 2021

Group Number: #105

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Kieran Adams	52804085	n5o1b	kieranadamslink1@gmail.com
Parsa Derakhshani	70248448	r1t2b	parsader2021@gmail.com
Andy Li	92239219	x7q1b	alaricli99@gmail.com

2. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why



Updates to our ER diagram:

- We added ISA constraints (total disjoint to User ISA and partial disjoint to Animal ISA) to our ISA relationships

- We added ticketID attribute to our Ticket entity because it is a better primary key than seat #
- We added the genus attribute to our Animal entity to create a functional dependency of the form species determines genus.

3. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:

a.List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...))
b.Specify the primary key, candidate key, foreign keys, and other constraints that the table must maintain

Sponsor(<u>name</u>: char, sponsorAmount: int)
WildAnimal(**<u>name</u>**: char, **<u>species</u>**: char, **sponsorName**: char)

- sponsorName CHAR(20) NOT NULL
- FOREIGN KEY (name, species) REFERENCES Animal(name, species) ON DELETE CASCADE
- FOREIGN KEY (sponsorName) REFERENCES Sponsor (name) ON DELETE CASCADE

DomesticatedAnimal(**<u>name</u>**: char, **<u>species</u>**: char)

- FOREIGN KEY (name, species) REFERENCES Animal(name, species) ON DELETE CASCADE

AnimalProduct(**<u>animalName</u>**: char, **<u>animalSpecies</u>**: char, <u>productName</u>: char, price: int, **giftShopName**: char)

- FOREIGN KEY (animalName, animalSpecies) REFERENCES DomesticatedAnimal(name, species) ON DELETE CASCADE,
- FOREIGN KEY (giftShopName) REFERENCES GiftShop (name) ON DELETE CASCADE

GiftShop(<u>name</u>: char)

AnimalProductProcured(**<u>visitorID</u>**: int, **<u>animalName</u>**: char, **<u>animalSpecies</u>**: char, **<u>productName</u>**: char)

- FOREIGN KEY (visitorID) REFERENCES Visitor(ID) ON DELETE CASCADE
- FOREIGN KEY (animalName, animalSpecies, productName) REFERENCES AnimalProduct (animalName, animalSpecies, productName) ON DELETE CASCADE

Show(<u>name</u>: char, <u>**habitatName**</u>: char)

- FOREIGN KEY (habitatName) REFERENCES Habitat(name) ON DELETE CASCADE

Ticket(<u>ticketID</u>: int, seatNum: int, price: int, **showName**: char, **visitorID**: int)

- FOREIGN KEY (showName) REFERENCES Show (name) ON DELETE CASCADE
- FOREIGN KEY (visitorID) REFERENCES Visitor(ID) ON DELETE CASCADE
- showName NOT NULL

Visitor(<u>ID</u>: int)

Animal(<u>name</u>: char, <u>species</u>: char, **habitatName**: char, **primaryCarerID**: int, genus: char)

- habitatName char(20) NOT NULL,
- genus char(20) NOT NULL,
- FOREIGN KEY (habitatName) REFERENCES Habitat(name) ON DELETE CASCADE
- FOREIGN KEY (primaryCarerID) REFERENCES PrimaryCarer (id) ON DELETE CASCADE

Habitat(<u>name</u>: char, capacity: int)

- capacity int DEFAULT 0

PrimaryCarer(<u>id</u>: int)

4. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key). You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown.

Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK – in at least two relations, so that they are at most 2NF. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process.

Functional Dependencies

The name and species of a wild animal determines the sponsor

The ticket id determines the show name

The ticket id determines the visitor ID

The ticket id determines the seat#

The ticket's seat # determines the price

The animal product name determines the domesticated animal's name and species

The animal product's name determines the gift shops name

An animal's name and species determines its habitat name

An animal's name and species determines the primary carer

An animal's species determines the animal's genus

Relation 1:

- Animal(<u>name</u>: char, <u>species</u>: char, **habitatName**: char, **primaryCarerID**: int, genus: char)
- **Functional Dependencies**
  - name, species -> habitatName in BCNF
  - species -> genus violates BCNF
  - name, species -> primaryCarerID in BCNF
- **Closures**
  - name, species += {name, species, habitatName, primaryCarerID, genus}
  - species += {species, genus}
- **Decomposed Tables**
  - AnimalGenus(<u>species</u>: char, genus: char)
  - Animal(<u>name</u>: char, **<u>species</u>**: char, **habitatName**: char, **primaryCarerID**: int)

Relation 2:

- Ticket(<u>ticketID</u>: int, seatNum: int, price: int, **showName**: char, **visitorID**: int)
- **Functional Dependencies**
  - ticketID -> showName, visitorID, seat#
  - seat# -> price
- **Closures**
  - ticketID += {ticketID, showName, visitorID, seat#, price} satisfies BCNF
  - seat# += {price} violates BCNF
- **Decomposed Tables**
  - Ticket(<u>ticketID:</u> int, **seatNum**: int, **showName**, **visitorID**)
  - SeatBracket(<u>seatNum:</u> int, price: int)

5. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization.The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post normalization.

Sponsor(<u>name</u>: char, sponsorAmount: int)

WildAnimal(<u>**name**</u>: char, <u>**species**</u>: char, **sponsorName**: char)

- sponsorName CHAR(20) NOT NULL
- FOREIGN KEY (name, species) REFERENCES Animal(name, species) ON DELETE CASCADE
- FOREIGN KEY (sponsorName) REFERENCES Sponsor (name) ON DELETE CASCADE
- FOREIGN KEY (species) REFERENCES AnimalGenus(species) ON DELETE CASCADE

DomesticatedAnimal(<u>**name**</u>: char, <u>**species**</u>: char)

- FOREIGN KEY (name, species) REFERENCES Animal(name, species) ON DELETE CASCADE
- FOREIGN KEY (species) REFERENCES AnimalGenus(species) ON DELETE CASCADE

AnimalProduct(<u>**animalName**</u>: char, <u>**animalSpecies**</u>: char, <u>productName</u>: char, price: int, **giftShopName**: char)

- FOREIGN KEY (animalName, animalSpecies) REFERENCES DomesticatedAnimal(name, species) ON DELETE CASCADE,
- FOREIGN KEY (giftShopName) REFERENCES GiftShop (name) ON DELETE CASCADE

GiftShop(<u>name</u>: char)

AnimalProductProcured(<u>**visitorID**</u>: int, <u>**animalName**</u>: char, <u>**animalSpecies**</u>: char, <u>**productName**</u>: char)

- FOREIGN KEY (visitorID) REFERENCES Visitor(ID) ON DELETE CASCADE
- FOREIGN KEY (animalName, animalSpecies, productName) REFERENCES AnimalProduct (animalName, animalSpecies, productName) ON DELETE CASCADE

Show(<u>name</u>: char, <u>**habitatName**</u>: char)

- FOREIGN KEY (habitatName) REFERENCES Habitat(name) ON DELETE CASCADE

Ticket(<u>ticketID</u>: int, **seatNum**: int, **showName**: char, **visitorID**: int)

- FOREIGN KEY (showName) REFERENCES Show (name) ON DELETE CASCADE
- FOREIGN KEY (visitorID) REFERENCES Visitor(ID) ON DELETE CASCADE
- FOREIGN KEY (seatNum) REFERENCES SeatBracket(seatNum) ON DELETE CASCADE
- showName NOT NULL
- seatNum NOT NULL

SeatBracket(<u>seatNum:</u> int, price: int)

- price NOT NULL

Visitor(<u>ID</u>: int)

Animal(<u>name</u>: char, **<u>species</u>**: char, **habitatName**: char, **primaryCarerID**: int)

- habitatName char(20) NOT NULL,
- FOREIGN KEY (habitatName) REFERENCES Habitat(name) ON DELETE CASCADE
- FOREIGN KEY (species) REFERENCES AnimalGenus(species) ON DELETE CASCADE
- FOREIGN KEY (primaryCarerID) REFERENCES PrimaryCarer (id) ON DELETE CASCADE

AnimalGenus(<u>species</u>: char, genus: char)

- genus char(20) NOT NULL,

Habitat(<u>name</u>: char, capacity: int)

- capacity int DEFAULT 0

PrimaryCarer(<u>id</u>: int)

6. The SQL DDL to create all the tables in SQL. All primary keys and foreign keys must be declared appropriately. Code the SQL CREATE TABLE statements with the appropriate foreign keys, primary keys, UNIQUE constraints, etc

```
CREATE TABLE Sponsor (

name char(20) PRIMARY KEY,

amountSponsored int

);
```

```
CREATE TABLE WildAnimal (

name char(20),

species char(20),

sponsorName char(20) NOT NULL,

PRIMARY KEY (name, species)

FOREIGN KEY (name, species) REFERENCES Animal(name, species) ON DELETE CASCADE,

FOREIGN KEY (sponsorName) REFERENCES Sponsor (name) ON DELETE CASCADE

FOREIGN KEY (species) REFERENCES AnimalGenus(species) ON DELETE CASCADE

);

CREATE TABLE DomesticatedAnimal (

name char(20),

species char(20),

PRIMARY KEY (name, species),

FOREIGN KEY (name, species) REFERENCES Animal(name, species) ON DELETE CASCADE,

FOREIGN KEY (species) REFERENCES AnimalGenus(species) ON DELETE CASCADE

);

CREATE TABLE AnimalProduct (

animalName char(20),

animalSpecies char(20),

productName char(20),

price int,
giftShopName char(20),
PRIMARY KEY(animalName, animalSpecies, productName),
FOREIGN KEY (animalName, animalSpecies) REFERENCES DomesticatedAnimal(name, species) ON
DELETE CASCADE,
FOREIGN KEY (giftShopName) REFERENCES GiftShop (name) ON DELETE CASCADE
);
CREATE TABLE GiftShop (
name char(20) PRIMARY KEY
);
CREATE TABLE AnimalProductProcured (
visitorID int,
animalName char(20),
animalSpecies char(20),
productName char(20),
PRIMARY KEY (animalName, animalSpecies, productName),
FOREIGN KEY (visitorID) REFERENCES Visitor(ID) ON DELETE CASCADE,
FOREIGN KEY (animalName, animalSpecies, productName) REFERENCES AnimalProduct
(animalName, animalSpecies, productName) ON DELETE CASCADE
);
CREATE TABLE Show (
name char(50),
habitatName char(20),
PRIMARY KEY (name, habitatName),

FOREIGN KEY (habitatName) REFERENCES Habitat(name) ON DELETE CASCADE

);

CREATE TABLE Ticket (

ticketID int PRIMARY KEY,

seatNum int,

showName char(50) NOT NULL,

visitorID int,

FOREIGN KEY (seatNum) REFERENCES SeatBracket(seatNum) ON DELETE CASCADE,

FOREIGN KEY (showName) REFERENCES Show (name) ON DELETE CASCADE,

FOREIGN KEY (visitorID) REFERENCES Visitor(ID) ON DELETE CASCADE

);

CREATE TABLE SeatBracket(

seatNum INT PRIMARY KEY,

price INT)


CREATE TABLE Visitor (

ID int PRIMARY KEY,

);

CREATE TABLE Animal (

name char(20),
species char(20),

habitatName char(20) NOT NULL,

primaryCarerID char(20),

PRIMARY KEY (name, species),

FOREIGN KEY (habitatName) REFERENCES Habitat(name) ON DELETE CASCADE,

FOREIGN KEY (species) REFERENCES AnimalGenus(species) ON DELETE CASCADE,

FOREIGN KEY (primaryCarerID) REFERENCES PrimaryCarer (id) ON DELETE CASCADE

);

CREATE TABLE AnimalGenus (

species char(20) PRIMARY KEY,

genus char(20) NOT NULL

);

CREATE TABLE Habitat (

name char(20) PRIMARY KEY,

capacity int DEFAULT 0

);

CREATE TABLE PrimaryCarer (

id int PRIMARY KEY

);
```

7. Populate each table with at least 5 tuples, and probably more so that you can issue meaningful Group By queries later on. Show the instance of each relation after inserting the tuples.

We recommend writing INSERT statements and submitting a .SQL file. However, screenshots of an excel tabular like format are accepted as well.

```
INSERT into Sponsor values
(“Timmy Rich”, 550);
INSERT into Sponsor values
(“Rachel Richer”, 750);
INSERT into Sponsor values
(“Gary Supericher”, 1450);
INSERT into Sponsor values
(“Melinda Ubericher”, 5000);
INSERT into Sponsor values
(“Jeffrey Bezos”, 50000);

INSERT into WildAnimal values
(“Timmy”, “Tiger”, “Timmy Rich”);
INSERT into WildAnimal values
(“Matilda”, “Moose”, “Melinda Ubericher”);
INSERT into WildAnimal values
(“Giovanni”, “Giraffe”, “Gary Supericher”);
INSERT into WildAnimal values
(“Rory”, “ Rhinoceros”, “Rachel Richer”);
INSERT into WildAnimal values
(“Juanita”, “ Jaguar”, “Jeffrey Bezos”);

INSERT into DomesticatedAnimal values
(“Romelu”, “Rooster”);
INSERT into DomesticatedAnimal values
(“Holly”, “Hen”);
INSERT into DomesticatedAnimal values
(“Helen”, “Hen”);
INSERT into DomesticatedAnimal values
(“Carrey”, “Cow”);
INSERT into DomesticatedAnimal values
```

```
("Grace", "Goat");
INSERT into DomesticatedAnimal values
("Peppa", "Pig");

INSERT into Animal values
("Timmy", "Tiger", "Tiger Enclosure", 56789);
INSERT into Animal values
("Matilda", "Moose", "Deers", 56783);
INSERT into Animal values
("Giovanni", "Giraffe", "Tall", 56783);
INSERT into Animal values
("Rory", " Rhinoceros", "Chunky", 56783);
INSERT into Animal values
("Juanita", " Jaguar", "Jaguar VIP Room", 56789);
INSERT into Animal values
("Romelu", "Rooster", "Chicken Coop", null);
INSERT into Animal values
("Holly", "Hen", "Chicken Coop", null);
INSERT into Animal values
("Helen", "Hen", "Chicken Coop", null);
INSERT into Animal values ,
("Carrey", "Cow", "Chicken Coop", null);
INSERT into Animal values
("Grace", "Goat", "Farm", null);
INSERT into Animal values
("Peppa", "Pig", "Pig Pen", null);
INSERT into Animal values
("Sammy", "Squirrel", "Rehab Center", 56789);

INSERT into AnimalProduct values
("Holly", "Hen","Eggs", 10, "Small Chicken Shop");
INSERT into AnimalProduct values
("Helen", "Hen","Eggs", 15, "Large Chicken Shop");
INSERT into AnimalProduct values
```

```
(“Grace”, “Goat”,“Cheese”, 20, “Goat Shop”);
INSERT into AnimalProduct values
(“Carrey”, “Cow”,“Yogurt”, 15, “Cow Shop”);
INSERT into AnimalProduct values
(“Carrey”, “Cow”,“Cheese”, 25, “Cow Shop”);

INSERT into GiftShopvalues
(“Cow Shop”);
INSERT into GiftShopvalues
(“Goat Shop”);
INSERT into GiftShopvalues
(“Pig Shop”);
INSERT into GiftShopvalues
(“Small Chicken Shop”);
INSERT into GiftShopvalues
(“Large Chicken Shop”);

INSERT into AnimalProductProcured
(12345,  “Grace”, “Goat”,“Cheese”);
INSERT into AnimalProductProcured
(12345,  “Carrey”, “Cow”,“Cheese”);
INSERT into AnimalProductProcured
(12346,  “Carrey”, “Cow”,“Cheese”);
INSERT into AnimalProductProcured
(12347,  “Carrey”, “Cow”,“Yogurt”);
INSERT into AnimalProductProcured
(12348,  “Holly”, “Hen”,“Eggs”);

INSERT into Show
(“Jaguar VIP Experience”,  “Jaguar VIP Room”);
INSERT into Show
(“Chicken Dance”, “Chicken Coop”);
INSERT into Show
(“Peppa Pig Concert”, “Pig Pen”);
```

```
INSERT into Show
(“Barnyard Tour”, “Cow Barn”);
INSERT into Show
(“Timmy The Tiger Stand Up Routine”, “Tiger Enclosure”);

INSERT into Ticket
(109, 9, “Jaguar VIP Experience”, 12345);
INSERT into Ticket
(237, 37, “Chicken Dance”, 12346);
INSERT into Ticket
(350, 50, “Peppa Pig Concert”, 12347);
INSERT into Ticket
(444, 44, “Barnyard Tour”, 12348);
INSERT into Ticket
(523, 23, “Timmy The Tiger Stand Up Routine”, 12349);

INSERT into SeatBracket
(9, 100);
INSERT into SeatBracket
(23, 70);
INSERT into SeatBracket
(37, 40);
INSERT into SeatBracket
(44, 20);
INSERT into SeatBracket
(50, 20);

INSERT into Visitor
(12345);
INSERT into Visitor
(12346);
INSERT into Visitor
(12347);
INSERT into Visitor
```

```
(12348);
INSERT into Visitor
(12349);

INSERT into Habitat
(“Chicken Coop”, 50);
INSERT into Habitat
(“Pig Pen”, 99);
INSERT into Habitat
(“Tiger Enclosure”, 50);
INSERT into Habitat
(“Jaguar VIP Room”, 10);
INSERT into Habitat
(“Cow Barn”, 75);
INSERT into Habitat
(“Rehab Center”, 75);
INSERT into Habitat
(“Tall”, 50);
INSERT into Habitat
(“Chunky”, 75);
INSERT into Habitat
(“Farm”, 99);

INSERT into PrimaryCarer
(56789)
INSERT into PrimaryCarer
(56780);
INSERT into PrimaryCarer
(56781);
INSERT into PrimaryCarer
(56782);
INSERT into PrimaryCarer
(56783);
```

INSERT into AnimalGenus

(“Tiger”, “Panthera”);

INSERT into AnimalGenus

(“Moose”, “Alces”);

INSERT into AnimalGenus

(“Giraffe”, “Giraffa”);

INSERT into AnimalGenus

(“Rhinoceros”, “Linnaeus”);

INSERT into AnimalGenus

(“Jaguar”, “Panthera”);

INSERT into AnimalGenus

(“Rooster”, “Junglefowl”);

INSERT into AnimalGenus

(“Hen”, “Junglefowl”);

INSERT into AnimalGenus

(“Cow”, “Bos”);

INSERT into AnimalGenus

(“Goat”, “Capra”);

INSERT into AnimalGenus

(“Pig”, “Sus”);

8. As much as possible, write a list of all the variety of queries that are proposed for your application, in plain English. For example, “Insertion: Add a customer to the supermarket membership list.” There may be some concepts that we have not yet covered by the time that the milestone is due. That’s okay. For example, if we have not covered projection by a week before the milestone is due, just look at it to get an idea of what is coming and write either something to the best of your ability or say “we have not covered projection yet, so we cannot answer this question.”

- **a. See milestone 5 of this document to see a list of all the types of required queries.**
    - **i. You only need to list the queries you plan to use to fulfill the requirements in milestone 5 and not every query for the application**
- **b. Note that you cannot “double dip” on a query (i.e., the same query cannot be used as an answer for multiple categories).**

c. **Note that our goal for this is to make sure that your schema is complex enough to ask the required questions over. You can certainly change them later.**

Insertion: Add a Visitor to the user list

Delete: Delete a Visitor from the user list

Update: Update the name of a show

Select: Select all tigers with the name “Timmy”

Projection: Project all sponsors who have contributed over $500

Join: Link Animal Products with Ticket to determine revenue

Aggregation with GROUP BY: Find the cheapest item in the Gift Shop

Aggregation with Having: Find all the visitors who have went to a Show

Nested Aggregation with Group by: we have not covered this yet, so we cannot answer this question

Division: we have not covered this yet, so we cannot answer this question