CS-336 Jurrasic Park Database for Summer 2015

Jessica Ackerman
CS 336 Rutgers University
Email: jessica.ackerman@rutgers.edu

Karin Kuo
CS 336 Rutgers University
Email: karin.kuo@rutgers.edu

Alec Donovan
CS 336 Rutgers University
Email: alec.donovan@rutgers.edu

***Abstract*— We will be creating the database required for running a zoo, more specifically, the zoo in question will be Jurrassic Park. Part of the database will be devoted to information on our dinosaurs: species, health, age, vaccinations, locations and enclosures, etc. Another part of the database will be devoted to employee information:shifts and payroll.**

I. Project Description

Insert your project here.You can refer to the Abstract of the 336ProjectDescriptionSummer2015 LaTeX file as an example. The project has four stages, as follows.

A. Stage1 - The Requirement Gathering Stage.

- The general system description: We will be creating the database required for running a zoo, more specifically, the zoo in question will be Jurassic Park. Part of the database will be devoted to information on our dinosaurs. Another part of the database will devoted to employees of the park such as veterinarians, dinosaur experts, and cashiers. We will also keep a table of guests who spectate the park.
- The 4 types of users (grouped by their data access/update rights): Please insert the users types in here, as follows:
  Tourist
  - The tourist's access rights: The tourists have limited access to certain information in the park database. They can view information about dinosaurs, tours, and concessions. However, they cannot edit certain data within other tables.
  - The real world scenarios:
    * Scenario 1 description: In scenario 1, a tourist would like to access information about a particular dinosaur. The tourist can go to an information terminal and sign in using their tourist id. They can then search the dinosaur species and learn about its diet.
    * System Data Input for Scenario1: tid, species
    * Input Data Types for Scenario1: int, char
    * System Data Output for Scenario1: species, diet
    * Output Data Types for Scenario1: char, char
    * Scenario 2 description: In scenario 2, a tourist would like to join a specific tour. To do so they access a terminal using their tid. After navigating to all possible tours for the day, tourists can choose a particular tour and join that tour.
    * System Data Input for Scenario 2: tid, time, date
    * Input Data Types for Scenario 2: int, int, int
    * System Data Output for Scenario 2: time, date
    * Output Data Types for Scenario 2: int, int

  Veterinarian
  - The veterinarian's access rights: The veterinarians have unlimited access to dinosaur health in the database. They can view information about health history and routine checkups for every dinosaur.
  - The real world scenarios:
    * Scenario 1 description: In scenario 1, a veterinarian makes routine checks on multiple dinosaurs throughout the course of the day. The dinosaurs he visits are T. Rex, Brontosaurus, and Raptor. After checking each dinosaur, he updates the health status of each dinosaur.
    * System Data Input for Scenario 1: ssn, did, health status, date
    * Input Data Types for Scenario 1: int, int, char, int
    * System Data Output for Scenario 1: null
    * Output Data Types for Scenario 1: null
    * Scenario 2 description: In scenario 2, a veterinarian needs to treat an injured or sick dinosaur. After performing necessary procedures the vet must access the database and log the date of when the dinosaur was treated so that other veterinarians know when that dinosaur had an emergency treatment.
    * System Data Input for Scenario 2: ssn, did, date
    * Input Data Types for Scenario 2: int, int, int
    * System Data Output for Scenario 2: null
    * Output Data Types for Scenario 2: null

  Expert
  - The expert's access rights: The dinosaur experts have access to dinosaur day to day schedules. They monitor eating habits and are responsible for dinosaur training.
  - The real world scenarios:
    * Scenario 1 description: In scenario 1, an expert decides to train a dinosaur to do a particular trick. They access dinosaur information in the database and endorses what tricks dinosaurs can perform. After training a dinosaur to speak, an expert can specify that in the table.
    * System Data Input for Scenario 1: ssn, did, trick
    * Input Data Types for Scenario 1: int, int, char

* System Data Output for Scenario 1: null
* Output Data Types for Scenario 1: null

* Scenario 2 description: In scenario 2, an expert should monitor the daily eating habits of each dinosaur. This means recording the amount of food each dinosaur is eating each day.
* System Data Input for Scenario 2: ssn, did, date, amount
* Input Data Types for Scenario 2: int, int, int, int
* System Data Output for Scenario 2: null
* Output Data Types for Scenario 2: null

Maintenance

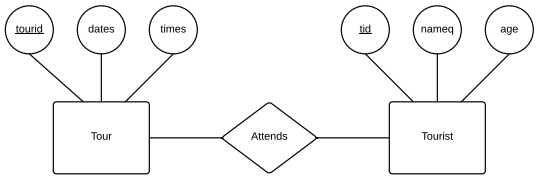
– Maintenance's access rights: Maintenance has access to equipment, vehicles, and all locations of the park.
– The real world scenarios:

* Scenario 1 description: In scenario 1, a maintenance worker fixes a fence and adds the serial number of the fence that is fixed to the relationship table.
* System Data Input for Scenario 1: ssn, serialIno
* Input Data Types for Scenario 1: int, int
* System Data Output for Scenario 1: null
* Output Data Types for Scenario 1: null

* Scenario 2 description: In scenario 2, a maintenance worker does a routine check on one of the vehicles used by tourists. After inspecting the vehicle the maintenance worker updates the status code to good. The system then verifies that the vehicle status has been modified.
* System Data Input for Scenario 2: ssn, vid, status
* Input Data Types for Scenario 2: int, int, char
* System Data Output for Scenario 2: vid, status
* Output Data Types for Scenario 2: int, char

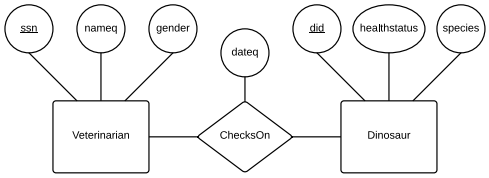
B. Stage2 - The Design Stage.

- The relation along with the attributes look like:



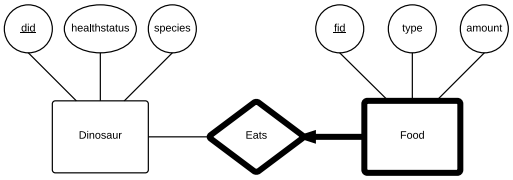
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Tourists attend tours. People who visit Jurassic park will take tours around the park. The tourist can see the times and dates of the tours around the park.

- The relation along with the attributes look like:



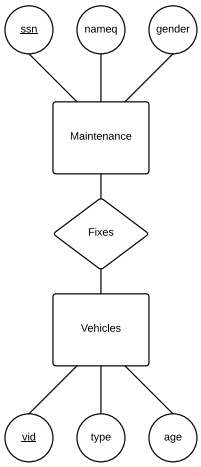
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): The Veterinarians check on the dinosaurs. The veterinarians perform check ups on the dinosaurs and records in the database the health status of the dinosaur and the date of the examination.

- The relation along with the attributes look like:



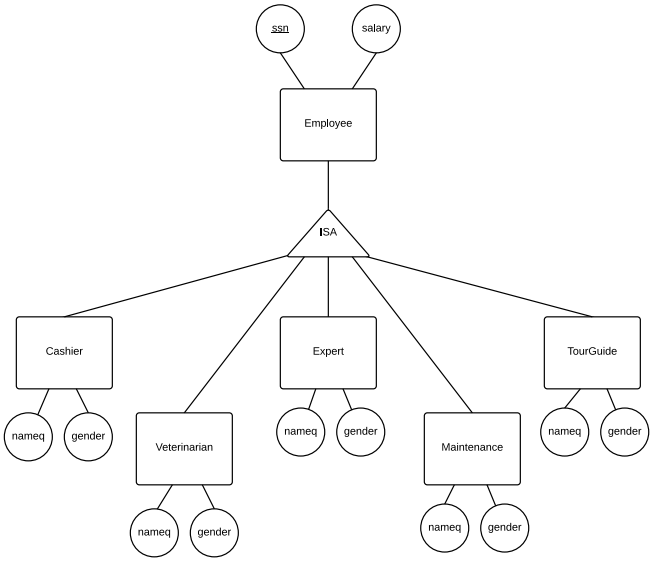
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): The dinosaurs eat food. The dinosaurs eat different amounts and different types of food depending on their species and this information is stored in the database and can be accessed by the expert that monitors that dinosaur. If the dinosaur no longer lives in the park, the park no longer needs to have its food.

- The relation along with the attributes look like:



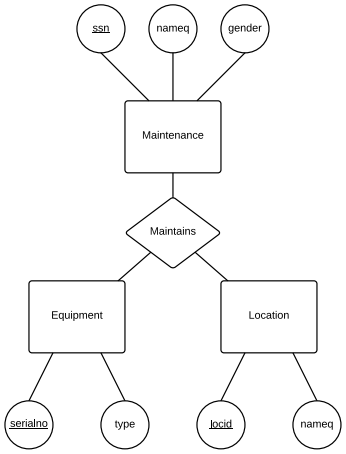
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): The maintenance workers fix the equipment. Part of the maintenance worker's jobs in the park is to fix the equipment which can be identified by their serial number and maintenance workers can access the database to see what equipment has been fixed and also to log equipment that they have fixed.

- The relation along with the attributes look like:



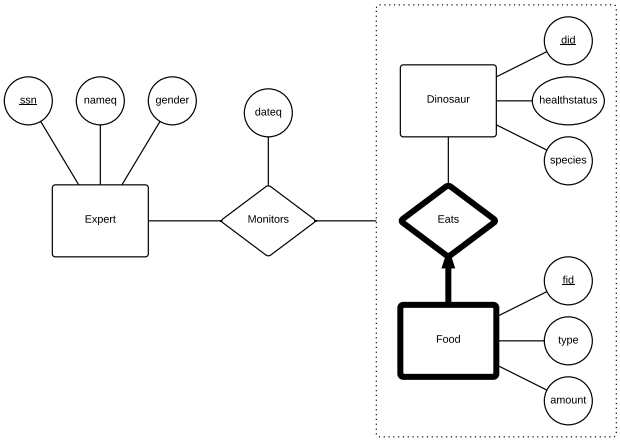
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Every employee is a veterinarian, expert, tour guide, cashier or maintenance worker. No employee of one type can be an employee of another type. Each employee is identified by their social security number, and has a salary and a name.

- The relation along with the attributes look like:



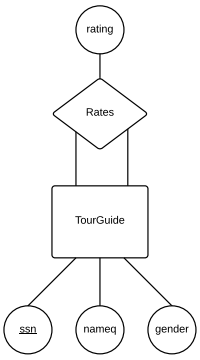
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Maintenance workers maintain vehicles, locations and equipment. Part of the maintenance worker's job is to make sure the vehicles, locations and equipment in the park are fully functional and operational and records the status in the database.

- The relation along with the attributes look like:



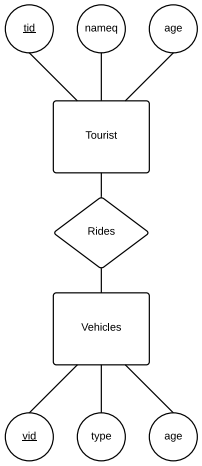
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): The experts monitor how much and what a dinosaur eats. Part of the job of the expert is to make sure the dinosaurs are eating properly and the expert can access the database to see how much the dinosaur eats and the health of the dinosaur.

- The relation along with the attributes look like:



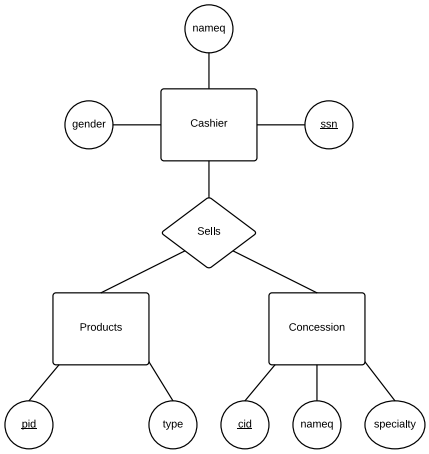
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Tour guides rate other tour guides and give them ratings. In order to make sure that the tours are the best they can be tour guides will give each other ratings and log into the database to record these ratings so that they can see their performance.

- The relation along with the attributes look like:



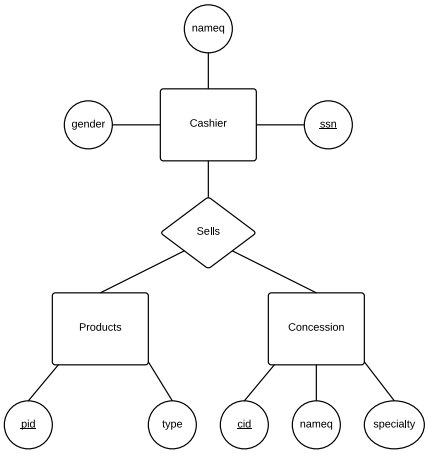
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Tourists ride vehicles. A tourist could log into the database to see what vehicles are available to them for transportation around the park and maintenance workers could log into the database to see which vehicles tourists are riding so that they can check those for repairs.

- The relation along with the attributes look like:



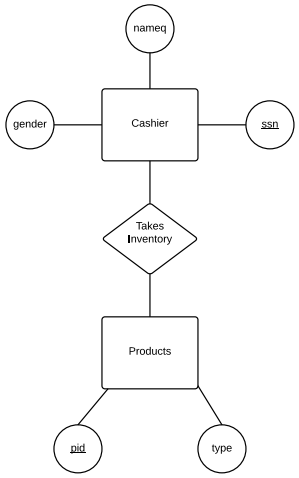
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Concessions sell products. A cashier or tourist could access the database so that they could see what products a concession sells.

- The relation along with the attributes look like:



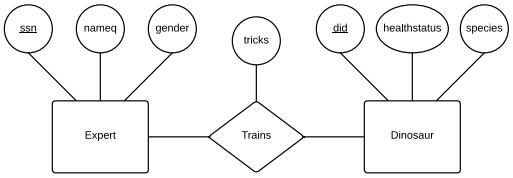
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Concessions sell products. A cashier or tourist could access the database so that they could see what products a concession sells.

- The relation along with the attributes look like:



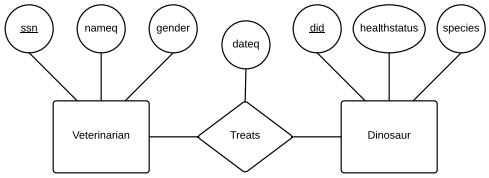
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Cashiers must keep an inventory of the products they stock. They have to access the database to keep a list of what products are currently in the concession.

- The relation along with the attributes look like:



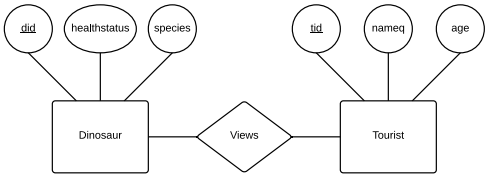
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Experts train dinosaurs to do tricks. Experts could access the database to see what dinosaurs know what tricks and which ones do not so that they could train them.

- The relation along with the attributes look like:



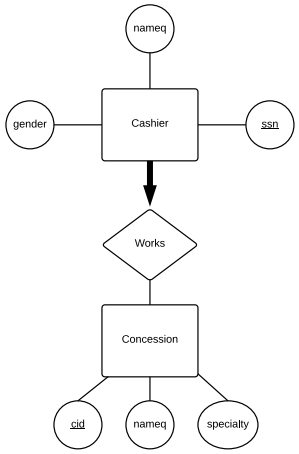
- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Veterinarians treat dinosaurs. If a dinosaur has a health problem after the health status has been changed by the veterinarian in the checks relation, the veterinarian can treat the dinosaur and log when the treatment was done.

- The relation along with the attributes look like:



- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Tourists can view dinosaurs. Tourists can access the database so that they can see the species and age of the dinosaurs and learn more about them like their diet.

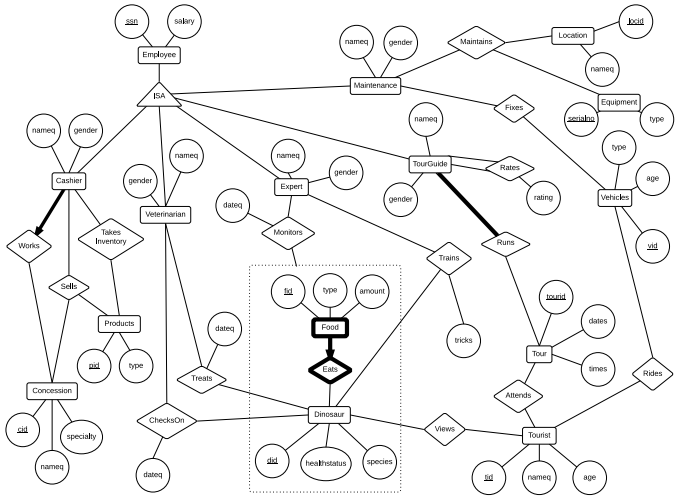
- The relation along with the attributes look like:



- The description relating this ER part (relation) as it corresponds to one or more user scenario(s): Cashiers work at concessions. Cashiers can access the database so that they can see which concession that they work at.

Please repeat that pattern for each relation.

- The ER diagram in its entirety:



- The ER diagram description (corresponding to the user scenarios): This ER diagram describes a database for a zoo. Inside the zoo there are dinosaurs, the employees who care for the dinosaurs, maintain the buildings and enclosures in the zoo, maintain the vehicles and equipment, run the concessions, give tours of the zoo and visit the zoo. Tourists can access the database to see times and dates and tours and to view information about the zoo's dinosaurs. The veterinarian accesses the database to record the health status of the zoo's dinosaurs and to record when the dinosaurs are treated for injury or illness. The expert trains and monitors the dinosaurs food and lifestyle. They can access the database to record what tricks the dinosaur knows and also to track the amount of food the dinosaur consumes and its health records. The cashier works at the concession and accesses the database to record the amount of inventory,or products, a certain concession has. The tour guide runs tours around the park and accesses the database to update times and dates of the tours and also to give other tour guides ratings. The maintenance workers maintain the functionality of the vehicles, equipment and locations in the park and can record the status of the fixes and needs for repair in the database.

Integrity Constraints:

- Overall:
  - Unique Constraint:
  - The primary keys for all relationships and entities are required to be not null. Entities are not allowed to have duplicate primary keys.
- The Employee ISA relationship:
  - Overlap Constraint:
  - The description and justification of the integrity constraint: Employees are not allowed to belong in multiple departments. An employee cannot be both a tour guide and a cashier.

  - Covering Constraint:
  - The description and justification of the integrity constraint: All employees must be either a tour guide, a cashier, a maintenance, an expert, or a veterinarian.
- Dinosaur Eats Food
  - Weak Entity Constraint:
  - The description and justification of the integrity constraint: Each Dinosaur can eat many types of food, and each type of food eaten by a dinosaur is assigned an id.
- Concessions:
  - Integrity Constraint: Each cashier works in exactly one concession stand.
  - The description and justification of the integrity constraint: Each cashier works at one and only one concession stand each day, which makes keeping track of registers and sales easier. Conversely, a concession stand could have zero cashiers (a vending machine), one cashier (a hot dog stand), or many cashiers (a busy Starbucks).
  - Integrity Constraint: Cashiers only take inventory of the products sold at their concession stands.
  - The description and justification of the integrity constraint: This is a practical constraint,
- Tours:
  - Integrity Constraint: Each tour has a maximum attendance of 30.
  - The description and justification of the integrity constraint: This is to prevent the tour guides from being overwhelmed, as well as keep the tours organized. This is not visible in the ER diagram, and not something that can be easily coded into the SQL tables.
- Tourist:
  - Integrity Constraint: Each tourist is logged into at most one vehicle at a time.
  - The description and justification of the integrity constraint: The "Rides" relation exists so the system can keep track of each tourist's current location for evacuation purposes. Therefore, as a person cannot be in two places at once, tourists can only be logged into one vehicle at a time.
- Maintenance:
  - Integrity Constraint: Maintenance checks up on all vehicles and equipment on a monthly basis. Maintenance fixes equipment and vehicles on an as needed basis.
- Veterinarians:
  - Integrity Constraint: Veterinarians check up on each dinosaur on a monthly basis. Veterinarians treat each dinosaur on an as needed basis

C. Stage3 - The Implementation Stage.

The "Employee" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Employee`(ssn int NOT NULL, salary int, Primary key(ssn));
Insert into `Employee` values (908,12000);
Insert into `Employee` values (909,15000);
Insert into `Employee` values(910,13000);
Insert into `Employee` values (911,10000);
Insert into `Employee` values (912,11000);
Insert into `Employee` values (913,10000);
Insert into `Employee` values (914,30000);
Insert into `Employee` values (915,34000);
Insert into `Employee` values (916,35000);
Insert into `Employee` values (917,40000);
Insert into `Employee` values (918,35000);
Insert into `Employee` values(919,31000);
Insert into `Employee` values(920,20000);
Insert into `Employee` values (921,22000);
Insert into `Employee` values (922,23000);
Insert into `Employee` values(923,15000);
Insert into `Employee` values (924,16000);
Insert into `Employee` values (925,12000);
mysql> Select * From Employee;
+-----+--------+
| ssn | salary |
+-----+--------+
| 908 |  12000 |
| 909 |  15000 |
| 910 |  13000 |
| 911 |  10000 |
| 912 |  11000 |
| 913 |  10000 |
| 914 |  30000 |
| 915 |  34000 |
| 916 |  35000 |
| 917 |  40000 |
| 918 |  35000 |
| 919 |  31000 |
| 920 |  20000 |
| 921 |  22000 |
| 922 |  23000 |
| 923 |  15000 |
| 924 |  16000 |
| 925 |  12000 |
+-----+--------+
```

```
mysql> Describe Employe;
ERROR 1146 (42S02): Table 'purple.Employe' doesn't exist
mysql> Describe Employee;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| ssn    | int(11) | NO   | PRI | NULL    |       |
| salary | int(11) | YES  |     | NULL    |       |
+--------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications:
  - The normalization step for the SQL table: Employee in 1 NF because each row only has one entry per attribute. All of the attributes are atomic. It is 2NF because it is in 1 NF and there is no partial key so no non-key depends on only part of the key. Each attribute depends on the whole key. The salary is dependant on the whole primary key, ssn. Employee is in 3NF because it is in 1NF and 2NF and because there are no transitive dependencies. There are only two attributes so it is not possible for there to be transitive dependencies.
- The Triggers used in the statement: Please insert the triggers are as follows:

```
mysql> CREATE TRIGGER `employeeraise`
    -> BEFORE INSERT ON `Employee`
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.salary < 20000
    -> THEN
    -> SET NEW.salary = NEW.salary*.1;
    -> END IF;
    -> END//
Query OK, 0 rows affected (0.13 sec)
```

  - The Trigger:
  - The Trigger Explanation/Justification: If an employee is making less than 20000 dollars he or she will get a 10 percent raise.
- The Data Range Constraints used in the statement:
  - The Data Range Constraint: The salary should be greater than 10,000 and no greater than 1,000,000.
  - The Data Range Constraint Explanation: The integer value for the salary should be held within realistc constraints.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user can not insert that an employee makes less than 10,000 dollars. This is the minimum salary.
  - The Data Access/update Constraint: Only the individual employees should be able to see only their own salary and ssn.
  - The Data Access/update Constraint Explanation: I It would be a security concern of employees if their ssn was available to every other employe.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A maintenance worker should not be able to see the ssn of a cashier.

The "Veterinarian" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Veterinarian`(ssn int NOT NULL, nameq CHAR(20), gender CHAR
(10),  Primary key(ssn), Foreign key(ssn) References Employee(ssn)ON DELETE
CASCADE);
Insert into `Veterinarian` values (914, "Cameron", "M");
Insert into `Veterinarian` values (915, "Sara", "F");
Insert into `Veterinarian` values (916, "Sean", "M");
```

```
mysql> Select * From Veterinarian;
+-----+---------+--------+
| ssn | nameq   | gender |
+-----+---------+--------+
| 914 | Cameron | M      |
| 915 | Sara    | F      |
| 916 | Sean    | M      |
+-----+---------+--------+
3 rows in set (0.00 sec)

mysql> Describe Veterinarian;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| ssn    | int(11)  | NO   | PRI | NULL    |       |
| nameq  | char(20) | YES  |     | NULL    |       |
| gender | char(10) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications:
  - The normalization step for the SQL table: Veterinarian is in 1NF because each attribute is atomic. There is one key and it points to only one entry per attribute. It is in 2NF because it is in 1NF and the attributes nameq and gender both depend on the entire key the ssn. It is in 3NF because it is in 1NF and 2NF and nameq does not depend on the gender and gender does not depend on nameq. There are no transitive dependencies.
- The Data Range Constraints used in the statement:
  - The Data Range Constraint: A user can only input M or F for the gender.
  - The Data Range Constraint Explanation: There are only two possible options in the context of this database.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to input "football" for gender.
  - The Data Access/update Constraint: Employees who access any of the employee types under the ISA relation can only see the name and gender of the employee. The ssn is hidden.
  - The Data Access/update Constraint Explanation: The ssn is hidden for the privacy of the employees.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A veterinarian looking up other veterinarians should not be able to access their social security numbers. This is a security concern.

The "Expert" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Expert`(ssn int NOT NULL, nameq CHAR(20), gender CHAR (10),
Primary key(ssn), Foreign key(ssn) References Employee(ssn)ON DELETE
CASCADE);
Insert into `Expert` values (908, "Sean", "M");
Insert into `Expert` values (909, "Chad", "M" );
Insert into `Expert` values(910, "Chris", "M");
Insert into `Expert` values (917, "Heather" , "F");
Insert into `Expert` values (918, "Desiree", "F");
Insert into `Expert` values(919, "Beth", "F");
```

```
mysql> Select * From Expert;
+-----+---------+--------+
| ssn | nameq   | gender |
+-----+---------+--------+
| 908 | Sean    | M      |
| 909 | Chad    | M      |
| 910 | Chris   | M      |
| 917 | Heather | F      |
| 918 | Desiree | F      |
| 919 | Beth    | F      |
+-----+---------+--------+
6 rows in set (0.00 sec)

mysql> Describe Expert;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| ssn    | int(11)  | NO   | PRI | NULL    |       |
| nameq  | char(20) | YES  |     | NULL    |       |
| gender | char(10) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Please insert the SQL Table, including data entries, in here.

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Expert is in 1NF because each attribute is atomic. There is one key and it points to only one entry per attribute. It is in 2NF because it is in 1NF and the attributes

nameq and gender both depend on the entire key the ssn. It is in 3NF because it is in 1NF and 2NF and nameq does not depend on the gender and gender does not depend on nameq. There are no transitive dependencies.

- The Data Range Constraint: A user can only input M or F for the gender.
- The Data Range Constraint Explanation: There are only two possible options in the context of this database.
- The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to input "football" for gender.
- The Data Access/update Constraint: Employees who access any of the employee types under the ISA relation can only see the name and gender of the employee. The ssn is hidden.
- The Data Access/update Constraint Explanation: The ssn is hidden for the privacy of the employees.
- The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A veterinarian looking up other veterinarians should not be able to access their social security numbers. This is a security concern.

The "Tour Guide" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `TourGuide`(ssn int NOT NULL, nameq CHAR(20), gender CHAR (10),
Primary key(ssn), Foreign key(ssn) References Employee(ssn)ON DELETE
CASCADE);
Insert into `TourGuide` values (923, "Kristin", "F");
Insert into `TourGuide ` values (924, "James", "M");
Insert into `TourGuide ` values (925, "Opal", "F");
mysql> Select * From TourGuide;
+-----+---------+--------+
| ssn | nameq   | gender |
+-----+---------+--------+
| 923 | Kristin | F      |
| 924 | James   | M      |
| 925 | Opal    | F      |
+-----+---------+--------+
3 rows in set (0.00 sec)

mysql> Describe TourGuide;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| ssn    | int(11)  | NO   | PRI | NULL    |       |
| nameq  | char(20) | YES  |     | NULL    |       |
| gender | char(10) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Please insert the SQL Table, including data entries, in here.

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: TourGuide is in 1NF because each attribute is atomic. There is one key and it points to only one entry per attribute. It is in 2NF because it is in 1NF and the attributes nameq and gender both depend on the entire key the ssn. It is in 3NF because it is in 1NF and 2NF and nameq does not depend on the gender and gender does not depend on nameq. There are no transitive dependencies.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.
- The Data Range Constraints used in the statement:
  - The Data Range Constraint: A user can only input M or F for the gender.
  - The Data Range Constraint Explanation: There are only two possible options in the context of this database.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to input "football" for gender.
  - The Data Access/update Constraint: Employees who access any of the employee types under the ISA relation can only see the name and gender of the employee. The ssn is hidden.
  - The Data Access/update Constraint Explanation: The ssn is hidden for the privacy of the employees.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A veterinarian looking up other veterinarians should not be able to access their social security numbers. This is a security concern.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Cashier" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Cashier`(ssn int NOT NULL, nameq CHAR(20), gender CHAR (10),
Primary key(ssn), Foreign key(ssn) References Employee(ssn) ON DELETE
CASCADE);
Insert into `Cashier` values (911, "Brittany", "F");
Insert into `Cashier` values (912, "Derin", "F");
Insert into `Cashier` values (913, "Derin", "F");
mysql> Select * From Cashier;
+-----+----------+--------+
| ssn | nameq    | gender |
+-----+----------+--------+
| 911 | Brittany | F      |
| 912 | Derin    | F      |
| 913 | Derin    | F      |
+-----+----------+--------+
3 rows in set (0.00 sec)

mysql> Describe Cashier;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| ssn    | int(11)  | NO   | PRI | NULL    |       |
| nameq  | char(20) | YES  |     | NULL    |       |
| gender | char(10) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications:
  - The normalization step for the SQL table: Cashier is in 1NF because each attribute is atomic. There is one key and it points to only one entry per attribute. It is in 2NF because it is in 1NF and the attributes nameq and gender both depend on the entire key the ssn. It is in 3NF because it is in 1NF and 2NF and nameq does not depend on the gender and gender

does not depend on nameq. There are no transitive dependencies.
  - The explanations/justification of the normalization step:

- The Data Range Constraints used in the statement:
  - The Data Range Constraint: A user can only input M or F for the gender.
  - The Data Range Constraint Explanation: There are only two possible options in the context of this database.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to input ”football” for gender.
  - The Data Access/update Constraint: Employees who access any of the employee types under the ISA relation can only see the name and gender of the employee. The ssn is hidden.
  - The Data Access/update Constraint Explanation: The ssn is hidden for the privacy of the employees.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A veterinarian looking up other veterinarians should not be able to access their social security numbers. This is a security concern.

The ”Maintenance” Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Maintenance`(ssn int NOT NULL, nameq CHAR(20), gender CHAR
(10), Primary key(ssn), Foreign key(ssn) References Employee(ssn) ON DELETE
CASCADE);
Insert into `Maintenance` values(920, "Laruen", "F");
Insert into `Maintenance` values (921, "Ian", "M");
Insert into `Maintenance` values (922, "Phillip", "M");
```

```
mysql> Select * From Maintenance;
+-----+---------+--------+
| ssn | nameq   | gender |
+-----+---------+--------+
| 920 | Laruen  | F      |
| 921 | Ian     | M      |
| 922 | Phillip | M      |
+-----+---------+--------+
3 rows in set (0.00 sec)

mysql> Describe Maintenance;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| ssn    | int(11)  | NO   | PRI | NULL    |       |
| nameq  | char(20) | YES  |     | NULL    |       |
| gender | char(10) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications:
  - The normalization step for the SQL table: Maintenance is in 1NF because each attribute is atomic. There is one key and it points to only one entry per attribute. It is in 2NF because it is in 1NF and the attributes nameq and gender both depend on the entire key the ssn. It is in 3NF because it is in 1NF and 2NF and nameq does not depend on the gender and gender does not depend on nameq. There are no transitive dependencies.
  - The explanations/justification of the normalization step:

- The Data Range Constraints used in the statement:
  - The Data Range Constraint: A user can only input M or F for the gender.
  - The Data Range Constraint Explanation: There are only two possible options in the context of this database.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to input ”football” for gender.
  - The Data Access/update Constraint: Employees who access any of the employee types under the ISA relation can only see the name and gender of the employee. The ssn is hidden.
  - The Data Access/update Constraint Explanation: The ssn is hidden for the privacy of the employees.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A veterinarian looking up other veterinarians should not be able to access their social security numbers. This is a security concern.

The ”Tourist” Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Tourist`(tid int NOT NULL, nameq CHAR(20), age int, CHECK (age >
0 AND age < 100));

Insert into `Tourist` values (415, "Ferris", 20);
Insert into `Tourist` values(416, "Cameron", 35);
Insert into `Tourist` values (417,"Sloan", 30);
Insert into `Tourist` values (418,"Heather",45);
Insert into `Tourist` values (419,"Joseph",8);
Insert into `Tourist` values (420,"Faith",55);
Insert into `Tourist` values (421,"Harry",10);
```

```
mysql> Select * From Tourist;
+-----+---------+------+
| tid | nameq   | age  |
+-----+---------+------+
| 415 | Ferris  |   20 |
| 416 | Cameron |   35 |
| 417 | Sloan   |   30 |
| 418 | Heather |   45 |
| 419 | Joseph  |    8 |
| 420 | Faith   |   55 |
| 421 | Harry   |   10 |
+-----+---------+------+
7 rows in set (0.00 sec)

mysql> Describe Tourist;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| tid   | int(11)  | NO   |     | NULL    |       |
| nameq | char(20) | YES  |     | NULL    |       |
| age   | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications:
  - The normalization step for the SQL table: Tourist is in 1 NF because the primary key only determines one value for each attribute type, nameq and age. Tourist is in 2NF because it is 1NF and because nameq and age depend on the entire primary key tid. It is also in 3NF because it is in 1NF, 2NF and because nameq does not depend on age and age does not depend on

nameq. There are no transitive dependencies.

- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:
  - The Trigger Explanation/Justification: This trigger occurs if a user attempts to input an invalid age.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): If a tourist attempts to insert that their age is 799, then this trigger will activate.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: No age can be greater than 0 or 120.
  - The Data Range Constraint Explanation: It is not possible for a person to be under the age of 0, and the oldest human is not older than 120 so no person inserted into this table should
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): For example if a Tourist was inserted into the system after entering the park and their inserted age was 779, then this would be outside of the possible range of ages for a human being and should not be allowed. If this value was to be inserted is would cause a trigger to activate.
  - The Data Access/update Constraint: Only Tourists can access this table.
  - The Data Access/update Constraint Explanation: No other entity in the data model needs to access this table so its access and updates are restricted to the Tourists who can update this table. Tourists themselves would not be able to change their tourist ID.
  - The Data Access/updaste Constraint Example(s) (in correlation with the user(s) interaction with the system): This access and update constraint would be executed through a view. As an example a Tourist could update their age and name in the system when they decide to visit the park so that they could sign up for tours and access dinosaur information.

The "Dinosaur" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Dinosaur`(did int NOT NULL, species CHAR(20), healthStatus
CHAR(20),Primary key(did), ADD CONSTRAINT chk_species CHECK ( (species =
'brontosaurus' OR species = 'trex' or species = 'raptor' or species =
'triceratops' or species = 'pterodactyl')), CONSTRAINT chk_healthStatus
CHECK (healthStatus = 'good' OR healthStatus = 'fair' OR healthStatus =
'poor'));

Insert Into `Dinosaur` values (1,"brontosaurus","fair");
Insert Into `Dinosaur`  values (2,"trex","good");
Insert Into `Dinosaur` values (3,"raptor","fair");

+--------------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> Select * From Dinosaur;
+-----+--------------+--------------+
| did | species      | healthStatus |
+-----+--------------+--------------+
|   1 | brontosaurus | fair         |
|   2 | trex         | good         |
|   3 | raptor       | fair         |
+-----+--------------+--------------+
3 rows in set (0.00 sec)

mysql> Describe Dinosaur;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| did          | int(11)  | NO   | PRI | NULL    |       |
| species      | char(20) | YES  |     | NULL    |       |
| healthStatus | char(20) | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications:
  - The normalization step for the SQL table: Dinosaurs is in 1 NF because each attribute depends on the key and the attributes do not have more than one value per entry. It is 2 NF because it is 1NF species and diet is determined by the entire key, did. There is no partial key. It is 3 NF because it is 1NF and 2NF and because there are no transitive dependencies. Species does not determine diet, and species does not determine diet. **********
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:
  - The Trigger Explanation/Justification: Please insert the Trigger explanation/justification in here.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): Please insert the Trigger examples in here.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: There are no data range constraints for the dinosaur table.
  - The Data Range Constraint Explanation: Any type of dinosaur can be inserted and any type of diet can be inserted.
  - The Data Access/update Constraint: Any tourist or employee can access the dinosaur table but only experts or veterinarians can update the table. Also

users should not be able to insert nonsense into the species value of the table.

- The Data Access/update Constraint Explanation: Tourists and any type of employee other than experts and veterinarians who works directly with the dinosaurs should not be able to insert dinosaurs into the table. A user should also not be allowed to insert a value in the species attribute that is not a species of dinosaur. This is enforced through an assertion that checks if the value inserted is one of the allowed values.
- The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): Access to the update the dinosaur table would be restricted to veterinarians and experts and no user would be able to see the dinosaur ID which would be enforced through the view. A trigger would be activated for any user other than the expert or veterinarian updating the table.

Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Food" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Food`(fid int NOT NULL, type CHAR(20), amount int, did int NOT
NULL, Primary key(fid, did), Foreign key(did) References Dinosaur(did),
CONSTRAINT chk_type CHECK ( type = 'hay' OR type = 'sheep' OR type = 'goats'
OR type = 'bugs' OR type = 'fish'), CHECK (amount > 0 AND amount < 100)ON
DELETE CASCADE);
Insert into `Food` values (645,"hay",70, 1);
Insert into `Food` values (646,"sheep",7, 2);
Insert into `Food` values (647,"pigs",2, 3);
```

```
mysql> Select * From Food;
+-----+-------+--------+-----+
| fid | type  | amount | did |
+-----+-------+--------+-----+
| 645 | hay   |     70 |   1 |
| 646 | sheep |      7 |   2 |
| 647 | pigs  |      2 |   3 |
+-----+-------+--------+-----+
3 rows in set (0.00 sec)

mysql> Describe Food;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| fid    | int(11)  | NO   | PRI | NULL    |       |
| type   | char(20) | YES  |     | NULL    |       |
| amount | int(11)  | YES  |     | NULL    |       |
| did    | int(11)  | NO   | PRI | NULL    |       |
+--------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Please insert the SQL Table, including data entries, in here.

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Eats is in 1 NF because each row does not have more than 1 value in each column. Each row is unique. It is 2 NF because amount is determined by the individual dinosaur's did and the fid. It is 3 NF because there are no transitive dependencies. There is only one non-key attribute
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:

```
mysql> CREATE TRIGGER `dinodiet`
    -> BEFORE INSERT ON `Food`
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.amount > 70
    -> THEN
    -> SET NEW.amount = NEW.amount*.5;
    -> END IF;
    -> END//
Query OK, 0 rows affected (0.63 sec)
```

  - The Trigger Explanation/Justification: If the dinosaur eats more than 70 of any type of food, the food will be cut in half so that the dinosaur is on a diet.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): An expert can input that a dinosaur eats 100 sheep but it will be on a diet and only eat 50 sheep.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The amount of food that could be inserted into the table should be between 0 - 100.
  - The Data Range Constraint Explanation: It should not be possible for a user to insert unreasonable amounts of food into the table.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): If an expert accesses the table to update the amount of food a dinosaur needs, he should not be able to insert that a dinosaur eats 100,000 sheep.
  - The Data Access/update Constraint: Only the expert can access this table.
  - The Data Access/update Constraint Explanation: Only the expert monitors what the dinosaur eats so only he can update and access a view of what each dinosaur eats and how much. Also users should not be able to insert values into the type attribute that are not food. This is enforced through an assertion that forces a check that the value inserted is from a list of foods.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): For example a cashier should not be able to go into the food table and update the table since he does not work in that department. This would be enforced through a view.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Tour" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Tour`(tourid int NOT NULL, times int, dates int, Primary
key(tourid), CHECK(time > 0 and time < 9999), CHECK(dates > 10120111 and
dates < 12312015)ON DELETE CASCADE ON UPDATE CASCADE;
Insert into `Tour` values(17,1030,03242015);
Insert into `Tour` values(18,1130,03252015);
Insert into `Tour` values(19,1230,03252015);
```

```
mysql> Select * From Tour;
+--------+-------+---------+
| tourid | times | dates   |
+--------+-------+---------+
|     17 |  1030 | 3242015 |
|     18 |  1130 | 3252015 |
|     19 |  1230 | 3252015 |
+--------+-------+---------+
3 rows in set (0.00 sec)

mysql> Describe Tour;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| tourid | int(11) | NO   | PRI | NULL    |       |
| times  | int(11) | YES  |     | NULL    |       |
| dates  | int(11) | YES  |     | NULL    |       |
+--------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

  Please insert the SQL Table, including data entries, in here.
- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Tours is in 1 NF because the primary key (tourid) uniquely identifies each tuple. It is in 2NF because time, the only non key attribute, depends on tourid the only primary key attribute. It is in 3NF because there are no transitive dependencies because there are only two attributes.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:

```
mysql> CREATE TRIGGER `runninglate`
    -> BEFORE INSERT ON `Tour`
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.dates = 3252015
    -> THEN
    -> SET NEW.times = NEW.times + 15;
    -> END IF;
    -> END//
Query OK, 0 rows affected (0.30 sec)
```

  - The Trigger Explanation/Justification: All tours where the date is 03252015 will be 15 minutes later
  - The Trigger Examples (in correlation with the user(s) interaction with the system): If a tour guide inserts a date that is 03252015 it will be fifteen minutes later than the time that was inserted.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The user should not be able to insert times larger than 4 digits and should not be able insert dates larger than 8 digits.
  - The Data Range Constraint Explanation: This is impossible because there is no time 13457, this does not make sense in the context with the attribute and the same goes for dates.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to insert a time for the tour to be at 567890, because this is not a time in the format of the times attribute.
  - The Data Access/update Constraint: Only tour guides can update this table and only tour guides and tourists can access this table.
  - The Data Access/update Constraint Explanation: Tour guides run the tours so they should be the only ones to update the times and dates of the tours. Tourists attend the tours so they should be able to see the dates and times.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A tour guide should not be able to insert a nonsense time or date because tourists would not be able to understand this information and no one would attend tours. This access and update constraint would be enforced through a view and trigger.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Equipment" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Equipment`(serialno int NOT NULL, type CHAR(20), Primary
key(serialno)) CHECK (serialno > 100,000 AND serialno < 109,999);
Insert into `Equipment` values(1023845, "fence");
Insert into `Equipment` values(1034578, "soda machine");
Insert into `Equipment` values(1056789, "toilet");
```

```
mysql> Select * From Equipment;
+----------+--------------+
| serialno | type         |
+----------+--------------+
|  1023845 | fence        |
|  1034578 | soda machine |
|  1056789 | toilet       |
+----------+--------------+
3 rows in set (0.00 sec)

mysql> Describe Equipment;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| serialno | int(11)  | NO   | PRI | NULL    |       |
| type     | char(20) | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

  Please insert the SQL Table, including data entries, in here.
- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Tours is in 1 NF because the primary key (serialno) uniquely identifies each tuple. It is in 2NF because time, the only non key attribute, depends on serialno the only primary key attribute. It is in 3NF because there are no transitive dependencies because there are only two attributes.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

Please repeat that pattern for every normalization step needed for that table.

- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:
  - The Trigger Explanation/Justification: if the serial number that is inserted does not follow the format then the trigger will be activated
  - The Trigger Examples (in correlation with the user(s) interaction with the system): If the serial number 1203981209381 is inserted, the trigger will be activated.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The only data range constraint is that serial no should be 7 digits and start with 100,000 so it can be between 100,000 and 109,999.
  - The Data Range Constraint Explanation: This is the format for serial numbers and a user should not be able to insert a nonsense serial number.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A maintenance worker should not be able to insert a nonsense serial number because then it would not correlate to any real world piece of equipment.
  - The Data Access/update Constraint: Only maintenance workers should be able to access and update this table.
  - The Data Access/update Constraint Explanation: Maintenance workers are the only ones that work with the equipment.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A cashier should not be able to insert a piece of equipment because he does not work with any equipment. This would be enforced through a trigger on the update.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Vehicles" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Vehicles`(vid int NOT NULL, type CHAR(20), age int, Primary
key(vid) CHECK (age > 0 AND age < 50));
Insert into `Vehicles` values (523,"A17",17);
Insert into `Vehicles` values (525,"B23",23);
Insert into `Vehicles` values (526,"C70",35);
```

```
mysql> Select * From Vehicles;
+-----+------+------+
| vid | type | age  |
+-----+------+------+
| 523 | A17  |   17 |
| 525 | B23  |   23 |
| 526 | C70  |   35 |
+-----+------+------+
3 rows in set (0.00 sec)

mysql> Describe Vehicles;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| vid   | int(11)  | NO   | PRI | NULL    |       |
| type  | char(20) | YES  |     | NULL    |       |
| age   | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

  Please insert the SQL Table, including data entries, in here.
- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Vehicle is in 1 NF because there is a primary key (transid) that can uniquely identify every tuple. Vehicle is in 2NF because all the non key attributes are determined by the whole key. Vehicle is in 3 NF because there are no transitive dependencies because there is only the primary key and one non key attribute.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:
  - The Trigger Explanation/Justification: A vehicle's age cannot be less than 0 or greater than 50 so if a value that are outside of these constraints then the trigger will be activated.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): If a maintenance work attempts to insert that vehicles is 70 years than the trigger will be activated.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: A user should not be able to insert an age below zero or over 50.
  - The Data Range Constraint Explanation: It is not possible for a vehicle to be under 0 years old and the park should not use any vehicle over 50 years old.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A maintenance worker should not be able insert a vehicle is 500 years old.

- The Data Access/update Constraint: Only maintenance workers should be able to access the
- The Data Access/update Constraint Explanation: Please insert the explanation in here.
- The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): Please insert the examples in here.

Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Location" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Location`(locid int NOT NULL, nameq CHAR(20), Primary
key(locid), CONSTRAINT chk_nameq CHECK (nameq = 'Building1' OR nameq =
'Building2' OR nameq = 'Building3' OR nameq = 'Building4' OR nameq =
'Building5')));
Insert into `Location` values (47,"Building1");
Insert into `Location` values (48,"Building2");
Insert into `Location` values (49,"Building3");
```

```
mysql> Select * From Location;
+-------+-----------+
| locid | nameq     |
+-------+-----------+
|    47 | Building1 |
|    48 | Building2 |
|    49 | Building3 |
+-------+-----------+
3 rows in set (0.00 sec)

mysql> Describe Location;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| locid | int(11)  | NO   | PRI | NULL    |       |
| nameq | char(20) | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Please insert the SQL Table, including data entries, in here.

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Locations is in 1 NF because the primary key (locid) uniquely identifies each tuple. It is in 2NF because time, the only non key attribute, depends on locid the only primary key attribute. It is in 3NF because there are no transitive dependencies because there are only two attributes.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:
  - The Trigger Explanation/Justification: If a maintenance worker attempts to insert a value for the name of building that does not exist within the constraints the trigger will activate.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): If a maintenance worker attempts to insert "football" into the table as name the trigger will activate.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: Only building names can be inserted into the name attribute of the location table.
  - The Data Range Constraint Explanation: A user should not be able to insert a nonsense value into the name of the location.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A maintenance worker can insert and update the location table and should not be able to insert "football" into the name of the location. This is enforced through an assertion that gives a list of values that can be inserted.
  - The Data Access/update Constraint: Only maintenance workers can access and update the location table.
  - The Data Access/update Constraint Explanation: The maintenance worker user are the only employee that works with locations.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A tourist should not be able to see the buildings or update the names of the buildings. This will be enforced through a trigger.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Concession" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Concession`(cid int NOT NULL, nameq CHAR(20), speciality
CHAR(20), Primary key(cid), CONSTRAINT chk_nameq (nameq = 'Dino Dogs' OR
nameq = 'Jurrasic Gifts' OR nameq = 'Prehistoric Pancakes' OR nameq =
'Rainforest Cafe' OR nameq = 'Souveniers Emporium'), CONSTRAINT
chk_speciality CHECK (speciality = 'food' OR speciality = 'gifts'))));

Insert into `Concession` values(789,"Dino Dogs", "food");
Insert into `Concession` values(790, "Jurrasic Gifts", "gifts");
Insert into `Concession` values(791, "Prehistoric Pancakes","food");
```

```
mysql> Select * From Concession;
+-----+----------------------+------------+
| cid | nameq                | speciality |
+-----+----------------------+------------+
| 789 | Dino Dogs            | food       |
| 790 | Jurrasic Gifts       | gifts      |
| 791 | Prehistoric Pancakes | food       |
+-----+----------------------+------------+
3 rows in set (0.00 sec)

mysql> Describe Concession;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| cid        | int(11)  | NO   | PRI | NULL    |       |
| nameq      | char(20) | YES  |     | NULL    |       |
| speciality | char(20) | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Please insert the SQL Table, including data entries, in here.

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.

  - The normalization step for the SQL table: Concession is in 1 NF because the primary key (cid) uniquely identifies each tuple. It is in 2NF because time, nameq and speciality do not form functional dependencies. It is in 3NF because there are no transitive dependencies.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger:
  - The Trigger Explanation/Justification: If a cashier attempts to insert a value outside of the constraints that does not make sense the the trigger will activate.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): If a cashier inputs "Steven" into the name column the trigger will activate.
  - The Trigger:
  - The Trigger Explanation/Justification: If a cashier attempts to insert a value outside of the constraints that does not make sense the the trigger will activate.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): If a cashier inputs "Steven" into the specialty column the trigger will activate.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: A constraint should be restricted in the values that can be inserted into name and speciality constraints so that nonsense names can not be inserted.
  - The Data Range Constraint Explanation: The names and specialties that are inserted into the table should only include names and specialties that actually exist in the park.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to insert "twenty" into the name of concession. This is enforced through assertions.
  - The Data Access/update Constraint: Only cashiers should be able to update and access the table. Tourists can access the table to see what vendors there are but their view is restricted to the name and speciality. This is enforced through views and triggers.
  - The Data Access/update Constraint Explanation: These are the users that work in the consessions so it would not make sense for another user to be able to update the table.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): An expert should not be able to change the names of the concessions or add new concessions to the table.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Product" Entity

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Products`(pid int NOT NULL, type CHAR(20), Primary key(pid),
CONSTRAINT chk_type CHECK (type = 'hot dogs' OR type = 'snow globes' OR type
= 'pancakes' OR type = 'posters' OR type = 'coffee')));
Insert into `Products` values(103, "hot dog");
Insert into `Products` values(104, "snow globes");
Insert into `Products` values(105, "pancakes");
```

```
mysql> Select * From Products;
+-----+-------------+
| pid | type        |
+-----+-------------+
| 103 | hot dog     |
| 104 | snow globes |
| 105 | pancakes    |
+-----+-------------+
3 rows in set (0.00 sec)

mysql> Describe Products;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| pid   | int(11)  | NO   | PRI | NULL    |       |
| type  | char(20) | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

  Please insert the SQL Table, including data entries, in here.
- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Products is in 1 NF because the primary key (pid) uniquely identifies each tuple. It is not in 2NF because type implies price. Please insert the explanations/justifications of the normalization step in here.
- The Data Range Constraints used in the statement:
  - The Data Range Constraint: Prices should be between 0 and 200 and the type of the products should be restricted to values that the concessions actually sell.
  - The Data Range Constraint Explanation: The price should be restricted to reasonable values and the table should restrict useres from inserting nonsense values.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): Please insert the examples in here.
  - The Data Access/update Constraint: Cashiers can update and access the table. Tourists can access the table but not see the pid.
  - The Data Access/update Constraint Explanation: Users that do not interact with the products or concessions should not have access to the table and

only cashiers who sell the products should be able to add or delete products.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A cashier should be able to insert hot dogs at price 7 in the table, tourists can see this but veterinarians should not be able to insert hot dogs at price 7 or access the products table. This is enforced through views and triggers.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Attends" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Attends`(tid int NOT NULL, tourid int, Primary key(tid,tourid),
Foreign key(tid) References Tourist(tid), Foreign key(tourid) References
Tour(tourid)ON DELETE CASCADE ON UPDATE CASCADE);
Insert into `Attends` values(415,17);
Insert into `Attends` values(416,18);
Insert into `Attends` values(417,19);
```

```
mysql> Select * From Attends;
+-----+--------+
| tid | tourid |
+-----+--------+
| 415 |     17 |
| 416 |     18 |
| 417 |     19 |
+-----+--------+
3 rows in set (0.00 sec)

mysql> Describe Attends;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| tid    | int(11) | NO   | PRI | NULL    |       |
| tourid | int(11) | NO   | PRI | 0       |       |
+--------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

  Please insert the SQL Table, including data entries, in here.
- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Attends is in 1 NF because the primary keys (tid and tourid) uniquely identify each tuple. It is in 2NF and 3NF because there are no non key attributes.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement: Please insert the triggers are as follows:
  - The Trigger: Please insert the first Trigger in here.
  - The Trigger Explanation/Justification: Please insert the Trigger explanation/justification in here.
  - The Trigger Examples (in correlation with the user(s) interaction with the system): Please insert the Trigger examples in here.

  Please repeat the pattern for every trigger in the CREATE TABLE SQL statement.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Access/update Constraint: Tour guides and tourists can view the table. Tourists can update the attends table so that they can sign up for the table. They should not be able to insert tid or tourid that do not exist.
  - The Data Access/update Constraint Explanation: Tourists need a way to sign up for the tour
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): Please insert the examples in here.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "ChecksOn" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `ChecksOn`(ssn int NOT NULL, did int NOT NULL, dateq int,
Primary key(ssn, did), Foreign key(ssn) References Veterinarian(ssn), Foreign
key(did) References Dinosaur(did), CONSTRAINT chk_heatlthStatus
CHECK(healthStatus = 'good' or healthStatus = 'fair' OR healthStatus =
'poor') CHECK (date > 1012011 and price < 12312015)));

Insert into `ChecksOn` values(914,1, 03242014);
Insert into `ChecksOn` values(915,2, 04252015);
Insert into `ChecksOn` values(916,3, 06262015);
```

```
mysql> Select * From ChecksOn;
+-----+-----+----------+
| ssn | did | dateq    |
+-----+-----+----------+
| 914 |   1 | 3242014 |
| 915 |   2 | 4252015 |
| 916 |   3 | 6262015 |
+-----+-----+----------+
3 rows in set (0.00 sec)

mysql> Describe ChecksOn;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| ssn   | int(11) | NO   | PRI | NULL    |       |
| did   | int(11) | NO   | PRI | NULL    |       |
| dateq | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications:
  - The normalization step for the SQL table: ChecksOn is in 1 NF because the primary key (ssn,did) uniquely identifies each tuple. It is in 2NF because time, all the non key attributes depend on the whole primary key. It is in 3NF because there are no transitive dependencies.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The date should not exceed 12312015 and should be greater than 1012011. The only healthStatus words the veterinarian should be able to input are good, fair and poor.
  - The Data Range Constraint Explanation: Users should not be able to insert dates that do not makes sense like 102319048012. This date does not exist. Users also should not be allowed to insert "football" into the healthStatus attribute.

- The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A veterinarian can insert 3262014 as a date and the healthStatus as good into the system after completing a check up on the dinosaur on that date.
- The Data Access/update Constraint: Only veterinarians should have access the health records of the dinosaurs.
- The Data Access/update Constraint Explanation: Only veterinarians can access the checkson table and its attributes to record the health of the dinosaurs.
- The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A tourist should not be able to insert a checkup on a dinosaur or edit the healthStatus of a dinosaur.

Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

-

The "Fixes" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Fixes`(vid int, ssn int, Primary key(vid,ssn), Foreign key(vid)
References Vehicles(vid)), Foreign key(ssn) References Maintenance(ssn)ON
DELETE CASCADE);
Insert into `Fixes` values(523, 920);
Insert into `Fixes` values(525, 921);
Insert into `Fixes` values(526, 922);

mysql> Select * From Fixes;
+-----+-----+
| vid | ssn |
+-----+-----+
| 523 | 920 |
| 525 | 921 |
| 526 | 922 |
+-----+-----+
3 rows in set (0.00 sec)

mysql> Describe Fixes;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| vid   | int(11) | NO   | PRI | 0       |       |
| ssn   | int(11) | NO   | PRI | 0       |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.21 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Fixes is in 1 NF because the primary key (ssn,serialno) uniquely identifies each tuple. It is in 2NF because time, all the non key attributes depend on the whole primary key. It is in 3NF because there are no transitive dependencies, there is only one non key attribute.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: Serial numbers need to be between 100,000 and 109,999. SSN should match up to employees that actually exist and are maintenance workers.
  - The Data Range Constraint Explanation: The serial numbers identify a piece of equipment and this is the format for how the serial numbers are put in and if the ssn does not match a person that exists or a maintenance worker than this relation does not make any sense.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A serial number of 102938019238 would not correspond to a real piece of equipment and these serial numbers should match up with the serial numbers in the equipment table as should the ssn of employees. A cashier would not fix a fence so it does not make sense to have their ssn in this table.
  - The Data Access/update Constraint: Only maintenance workers can access and update this table.
  - The Data Access/update Constraint Explanation: Other users who do not interact with equipment should not be able to make updates to what equipment was fixed and by whom.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A tourist should not be able to insert that a fence was fixed by a maintenance worker, they have no knowledge of this information.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

-

The "Maintains" Relation

- The SQL Table, including data entries and SQL statement to create the table [We had issues inputting the SQL for the ternary relationship "Maintains", so the table does not exist.]
- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Please insert the first normalization step for the table in here.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The SQL statement to create the table:
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: Please insert the first Data Range Constraint in here.
  - The Data Range Constraint Explanation: Please insert the explanation in here.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): Please insert the examples in here.

- The Data Access/update Constraint: Please insert the Data Access/update Constraint in here.
- The Data Access/update Constraint Explanation: Please insert the explanation in here.
- The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): Please insert the examples in here.

Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Monitors" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Monitors`(did int, ssn int NOT NULL, fid int NOT NULL, dateq
int, Primary key(did,ssn,fid), Foreign key(did) References Dinosaur(did),
Foreign key(ssn) References Expert(ssn), Foreign key(fid) References
Food(fid), CHECK(date > 10120111 and dates < 12312015) ON DELETE CASCADE );

Insert into `Monitors` values (1,908,645,03282013);
Insert into `Monitors` values (2,909,646,04292015);
Insert into `Monitors` values (3,910,647,05152015);
```

```
mysql> Select * From Monitors;
+-----+-----+-----+---------+
| did | ssn | fid | dateq   |
+-----+-----+-----+---------+
|   1 | 908 | 645 | 3282013 |
|   2 | 909 | 646 | 4292015 |
|   3 | 910 | 647 | 5152015 |
+-----+-----+-----+---------+
3 rows in set (0.00 sec)

mysql> Describe Monitors;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| did   | int(11) | NO   | PRI | 0       |       |
| ssn   | int(11) | NO   | PRI | NULL    |       |
| fid   | int(11) | NO   | PRI | NULL    |       |
| dateq | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Fixes is in 1 NF because the primary key (did,ssn,fid) uniquely identifies each tuple. It is in 2NF because time, all the non key attributes depend on the whole primary key. It is in 3NF because there are no transitive dependencies, there is only one non key attribute.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Triggers used in the statement:
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The date should not exceed 12312015 and should be greater than 1012011. All primary keys should match primary keys in the other tables.
  - The Data Range Constraint Explanation: Only real and possible dates should be inserted into the table.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A user should not be able to input that he monitored a dinosaur on some day in 2017, this has not happened yet.
  - The Data Access/update Constraint: Only experts can access and update this table.
  - The Data Access/update Constraint Explanation: Experts are the only ones who monitor the dinosaurs so no other user should be updating this table.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A cashier should not be able to record a date that the dinosaur was monitored.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Rates" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Rates`(mainSSN int NOT NULL, otherSSN int NOT NULL, rating int,
Primary Key(mainSSN,otherSSN),Foreign Key(mainSSN) References TourGuide(ssn),
Foreign key(otherSSN) References TourGuide(ssn), CHECK(rating > 0 and rating
< 10)ON DELETE CASCADE);
Insert into `Rates` values(923,924,5);
Insert into `Rates` values(923,925,8);
Insert into `Rates` values(924,925,7);
```

```
mysql> Select * From Rates;
+---------+----------+--------+
| mainSSN | otherSSN | rating |
+---------+----------+--------+
|     923 |      924 |      5 |
|     923 |      925 |      8 |
|     924 |      925 |      7 |
+---------+----------+--------+
3 rows in set (0.00 sec)

mysql> Describe Rates;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| mainSSN  | int(11) | NO   | PRI | NULL    |       |
| otherSSN | int(11) | NO   | PRI | NULL    |       |
| rating   | int(11) | YES  |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Fixes is in 1 NF because the primary key (ssn) uniquely identifies each tuple. It is in 2NF because rating, all the non key attributes depend on the whole primary key. It is in 3NF because there are no transitive dependencies, there is only one non key attribute.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: Ratings must be between 1 and 10.
  - The Data Range Constraint Explanation: The system for ratings at the park only exist on a scale between 1 and 10.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): PA

tour guide can not insert a 0 rating or a 20 rating for another tour guide.
  - The Data Access/update Constraint: Only tour guides can access and update the rates table.
  - The Data Access/update Constraint Explanation: Only tour guides can rate other tour guides.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A maintenance worker can not rate the tour guides.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Rides" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Rides`(tid int NOT NULL, vid int NOT NULL, Primary
key(tid,vid), Foreign key(tid) References Tourist(tid), Foreign key(vid)
References Vehicles(vid)ON DELETE CASCADE);
Insert into `Rides` values (415,523);
Insert into `Rides` values (416,525);
Insert into `Rides` values (417,526);
```

```
mysql> Select * From Rides;
+-----+-----+
| tid | vid |
+-----+-----+
| 415 | 523 |
| 416 | 525 |
| 417 | 526 |
+-----+-----+
3 rows in set (0.00 sec)

mysql> Describe Rides;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| tid   | int(11) | NO   | PRI | NULL    |       |
| vid   | int(11) | NO   | PRI | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Rides is in 1 NF because the primary key (tid,vid) uniquely identifies each tuple. It is in 2NF and 3NF because there are no non key attributes.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.
  - The Data Range Constraint: All primary keys in the Rides table should match the primary keys in the tourist and vehicles relations.
  - The Data Access/update Constraint: Only tourists and maintenance workers should be able to access the Rides table.
  - The Data Access/update Constraint Explanation: Tourists can update what vehicles they ride and maintenance workers can only view the table so as to see what vehicles need to maintained.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A tour guide should not be able to access the Rides table or change any of the tourists who rode different vehicles.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Runs" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Runs`(ssn int NOT NULL, tourid int NOT NULL, Primary
key(ssn,tourid), Foreign key(ssn) References TourGuide(ssn), Foreign
key(tourid) References(tourid)ON DELETE CASCADE);

Insert into `Runs` values(923,17);
Insert into `Runs` values (924,18);
Insert into `Runs` values(925,19);
```

```
mysql> Select * From Runs;
+-----+--------+
| ssn | tourid |
+-----+--------+
| 923 |     17 |
| 924 |     18 |
| 925 |     19 |
+-----+--------+
3 rows in set (0.00 sec)

mysql> Describe Runs;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| ssn    | int(11) | NO   | PRI | NULL    |       |
| tourid | int(11) | NO   | PRI | NULL    |       |
+--------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Rides is in 1 NF because the primary key (ssn,tourid) uniquely identifies each tuple. It is in 2NF and 3NF because there are no non key attributes.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The primary keys for Tour Guides and Tours should match the primary keys in Runs.
  - The Data Access/update Constraint: Only tour guides can update and access this table.
  - The Data Access/update Constraint Explanation: Tour Guides run tours so they should be the only ones to update this table.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A tour guide runs a tour and updates the table to specify which tour they run.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The "Sells" Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Sells`(pid int NOT NULL, cid int NOT NULL, ssn int, itemPrice
int, Primary key(pid,cid,ssn), Foreign key(ssn) References Cashier(ssn),
Foreign key(pid) References Products(pid), Foreign key(cid) References
Concession(cid), CHECK (price > 0 and price < 200)))ON DELETE CASCADE ON
UPDATE CASCADE);

Insert into `Sells` values(103,789, 911, 10);
Insert into `Sells` values(104,790, 912, 9);
Insert into `Sells` values(105,791,913, 12);
```

```
mysql> Select * From Sells;
+-----+-----+-----+-----------+
| pid | cid | ssn | itemPrice |
+-----+-----+-----+-----------+
| 103 | 789 | 911 |        10 |
| 104 | 790 | 912 |         9 |
| 105 | 791 | 913 |        12 |
+-----+-----+-----+-----------+
3 rows in set (0.00 sec)

mysql> Describe Sells;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| pid       | int(11) | NO   | PRI | NULL    |       |
| cid       | int(11) | NO   | PRI | NULL    |       |
| ssn       | int(11) | NO   | PRI | 0       |       |
| itemPrice | int(11) | YES  |     | NULL    |       |
+-----------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Sells is in 1NF because there is only one value in each attribute that functionally depend on the key. It is in 2NF because itemPrice depends on the product
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The primary keys in the sells table should match the primary keys in their referenced tables.
  - The Data Access/update Constraint: Only cashiers and tourists can access the table. Only cashiers can update the table.
  - The Data Access/update Constraint Explanation: The sells table involves what products a concession sells. This is determined by the cashier. Tourists can see what products and concession sells but not update the table.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): A Tourist can log on and see what products they can buy from certain locations. A cashier can update new products or delete products they no longer sell.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The ”Trains” Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Trains`(ssn int NOT NULL, did int NOT NULL, tricks
CHAR(20),Primary key(ssn,did),Foreign key(ssn) References Expert(ssn),
Foreign key(did) References Dinosaur(did), CONSTRAINT chk_tricks CHECK
(tricks = 'stay' OR tricks = 'sit' OR tricks = 'speak'));

Insert into `Trains` values(917,1,"sit");
Insert into `Trains` values(918,2,"stay");
Insert into `Trains` values(919,3,"speak");
```

```
mysql> Select * From Trains;
+-----+-----+--------+
| ssn | did | tricks |
+-----+-----+--------+
| 917 |   1 | sit    |
| 918 |   2 | stay   |
| 919 |   3 | speak  |
+-----+-----+--------+
3 rows in set (0.00 sec)

mysql> Describe Trains;
+--------+----------+------+-----+---------+-------+
| Field  | Type     | Null | Key | Default | Extra |
+--------+----------+------+-----+---------+-------+
| ssn    | int(11)  | NO   | PRI | NULL    |       |
| did    | int(11)  | NO   | PRI | NULL    |       |
| tricks | char(20) | YES  |     | NULL    |       |
+--------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Trains is in 1 NF because the primary key (did,ssn) uniquely identifies each tuple. It is in 2NF because time, the only non key attribute, depends on the whole primary key. It is in 3NF because there are no transitive dependencies because there is only one non key attribute.

  Please repeat that pattern for every normalization step needed for that table.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: A user should only be able to input certain values into tricks. The primary keys must match the primary keys that they reference.
  - The Data Range Constraint Explanation: A user should not be able to input nonsense into the trick attribute.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): The only values an expert should be able to input into tricks is ”sit”, ”stay” and ”speak”
  - The Data Access/update Constraint: Only experts can view and update the table.
  - The Data Access/update Constraint Explanation: The experts train the dinosaurs in tricks and no other entity does this so only the expert is allowed access.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): An expert trains a dinosaur on a new trick and logs into the database to update which dinosaur learned what trick and who taught it. A cashier does not have this information nor does he train dinosaurs.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The ”Treats” Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Treats`(ssn int NOT NULL, did int NOT NULL, dateq int, Primary
key(ssn, did), Foreign key(ssn) References Veterinarian(ssn), Foreign
key(did) References Dinosaur(did), CHECK(date > 1012011 and date < 12312015);

Insert into `Treats` values(914,1,03232011);
Insert into `Treats` values (915,2,04152012);
Insert into `Treats` values (916,3,06172014);
```

```
mysql> Select * From Treats;
+-----+-----+---------+
| ssn | did | dateq   |
+-----+-----+---------+
| 914 |   1 | 3232011 |
| 915 |   2 | 4152012 |
| 916 |   3 | 6172014 |
+-----+-----+---------+
3 rows in set (0.00 sec)

mysql> Describe Treats;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| ssn   | int(11) | NO   | PRI | NULL    |       |
| did   | int(11) | NO   | PRI | NULL    |       |
| dateq | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Trains is in 1 NF because the primary key (did,ssn) uniquely identifies each tuple. It is in 2NF because time, the only non key attribute, depends on the whole primary key. It is in 3NF because there are no transitive dependencies because there is only one non key attribute.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.

  Please repeat that pattern for every normalization step needed for that table.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: The date should not exceed 12312015 and should be greater than 1012011. All primary keys should match primary keys in the other tables.
  - The Data Range Constraint Explanation: Only real and possible dates should be inserted into the table.
  - The Data Range Constraint Example(s) (in correlation with the user(s) interaction with the system): A veterinarian can not insert a date that they checked the dinosaur if the date has not happened yet.
  - The Data Access/update Constraint: Only veterinarians can access and update the Treats table.
  - The Data Access/update Constraint Explanation: Only veterinarians can update and access this table because they are the only ones that treat the dinosaurs for health issues so they are the only ones with the information that would go into the table.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): After a veterinarian treats a dinosaur for a health problem they can record what dinosaur they treated and on what day.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The ”Views” Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
CREATE Table `Views` (did int NOT NULL, tid int NOT NULL, Primary
key(did,tid), Foreign key(did) References Dinosaur, Foreign key(tid)
References Tourist(tid)ON DELETE CASCADE);

Insert into `Views` values(1,415);
Insert into `Views` values(2,416);
Insert into `Views` values(3,417);
```

```
mysql> Select * From Views;
+-----+-----+
| did | tid |
+-----+-----+
|   1 | 415 |
|   2 | 416 |
|   3 | 417 |
+-----+-----+
3 rows in set (0.00 sec)

mysql> Describe Views;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| did   | int(11) | NO   | PRI | NULL    |       |
| tid   | int(11) | NO   | PRI | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Views is in 1 NF because the primary key (did,tid) uniquely identifies each tuple. It is in 2NF and 3NF because there are no non key attributes.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: All primary keys in views must match the primary keys in the tables they reference.
  - The Data Access/update Constraint: Only tourists, veterinarians, and experts can view the dinosaur information.
  - The Data Access/update Constraint Explanation: Only experts can update the table but tourists, veterinarians and experts can access the table. Only experts can update because they monitor the dinosaurs.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): There needs to be at least one user that can delete or add dinosaurs as they come into or leave the park.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

The ”Works” Relation

- The SQL Table, including data entries and SQL statement to create the table:

```
Create Table `Works`(cid int NOT NULL, ssn int NOT NULL, Primary
key(cid,ssn), Foreign key(cid) References Concession(cid), Foreign key(ssn)
References Cashier(ssn)ON DELETE CASCADE ON UPDATE CASCADE);


Insert into `Works` values(789,911);
Insert into `Works` values(790,912);
Insert into `Works` values(791,913);
```

```
mysql> Select * From Works;
+-----+-----+
| cid | ssn |
+-----+-----+
| 789 | 911 |
| 790 | 912 |
| 791 | 913 |
+-----+-----+
3 rows in set (0.00 sec)

mysql> Describe Works;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| cid   | int(11) | NO   | PRI | NULL    |       |
| ssn   | int(11) | NO   | PRI | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

- The normalization steps for each table, along with explanations/justifications: The normalization steps should be inserted as follows.
  - The normalization step for the SQL table: Views is in 1 NF because the primary key (cid,ssn) uniquely identifies each tuple. It is in 2NF and 3NF because there are no non key attributes.
  - The explanations/justification of the normalization step: Please insert the explanations/justifications of the normalization step in here.
- The Data Range Constraints used in the statement: Please insert the Data Range Constraints as follows:
  - The Data Range Constraint: All primary keys in views must match the primary keys in the tables they reference.
  - The Data Access/update Constraint: Only cashiers can access and update this table.
  - The Data Access/update Constraint Explanation: Only cashiers work at the concessions so only they can update the table.
  - The Data Access/update Constraint Example(s) (in correlation with the user(s) interaction with the system): Cashiers can update which concession that they work at or delete concessions that they no longer work at.

  Repeat that pattern for every constraint in the CREATE TABLE SQL statement.

Repeat that pattern for every table created.

D. Stage4 - User Interface.

Describing of a User Interface (UI) to the system along with the related information that will be shown on each screen (How users will query the tables and view the query results). The emphasis should be placed on the process required for a user in order to meet a particular information need, in a user-friendly manner. The deliverables for this stage include the following items (Please refer to the original Project Description for more details):

- The SQL statements used to query the data.
- The error messages that will pop-up when users access and/or updates are denied.
- The error messages corresponding to the integrity constraints violations.
- The error messages corresponding to the data range constraints violations.
- The header of the views created in order to facilitate data accesses, according to users' needs.
- Each view created must be justified. Any triggers built upon those views should be explained and justified as well. At least one view should be created and justified for the project.

Please insert your deliverables for Stage4 as follows:

- The the first SQL statement used to query the data:

```
ERROR 1054 (42S22): Unknown column 'T.namesq' in 'field list'
mysql> SELECT T.nameq
    -> FROM Employee E, TourGuide T, Rates R
    -> Where E.ssn = T.ssn AND otherSSN = T.ssn AND rating > 7;
    -> //
+-------+
| nameq |
+-------+
| Opal  |
+-------+
1 row in set (0.01 sec)
```

  - The name of the first table taking part in the query: Employees E
  - The header of the table (all attributes): The attributes of Employees is ssn and salary.
  - The attributes of the table taking part in the query: The ssn is connected to the Tour Guide table in this query.
  - The name of the first table taking part in the query: Tour Guide T
  - The header of the table (all attributes): The attributes are ssn, nameq and gender.
  - The attributes of the table taking part in the query: The ssn and the nameq are taking part in this query.
  - The name of the first table taking part in the query: Tour Guide T
  - The header of the table (all attributes): The attributes are ssn, nameq and gender.
  - The attributes of the table taking part in the query: The ssn and the nameq are taking part in this query.
  - The name of the first table taking part in the query: Rates R
  - The header of the table (all attributes): The attributes are mainSSN, other SSN and rating.
  - The attributes of the table taking part in the query: The rating and otherSSN are taking part in this query.
- The the second SQL statement used to query the data:

```
mysql> SELECT V.nameq
    -> FROM Veterinarian V, ChecksOn C, Dinosaur D
    -> WHERE V.ssn = C.ssn AND D.did = C.did AND D.species = "raptor";
+-------+
| nameq |
+-------+
| Sean  |
+-------+
```

  - The name of the first table taking part in the query: Employees E
  - The header of the table (all attributes): The attributes of Employees is ssn and salary.
  - The attributes of the table taking part in the query: The ssn is connected to the Tour Guide table in this query.
  - The name of the first table taking part in the query: Tour Guide T
  - The header of the table (all attributes): The attributes are ssn, nameq and gender.
  - The attributes of the table taking part in the query: The ssn and the nameq are taking part in this query.
  - The name of the first table taking part in the query: Tour Guide T
  - The header of the table (all attributes): The attributes are ssn, nameq and gender.
  - The attributes of the table taking part in the query: The ssn and the nameq are taking part in this query.
  - The name of the first table taking part in the query: Rates R
  - The header of the table (all attributes): The attributes are mainSSN, other SSN and rating.
  - The attributes of the table taking part in the query: The rating and otherSSN are taking part in this query.
- The error messages popping-up when users access and/or updates are denied (along with explanations and examples):
  - The error message: Please insert the error message in here.
  - The error message explanation (upon which violation does it take place): Please insert the error message explanation in here.
  - The error message example according to user(s) scenario(s): Please insert the error message example in here.
- The error messages corresponding to the integrity constraints violations (along with explanations and examples).
  - The error message: When an integrity constraint is violated then mysql will output an error message that specifies the type of error that occured becasue of the constraint violation.
  - The error message explanation (upon which violation does it take place): If a user attempts to insert an entry with a primary key that is already in the table than mysql will output an error message that specifies that this is a duplicate entry and can not be inserted into the table.
- The error messages corresponding to the data range constraints violations (along with explanation).
  - The error message: "BLANK is out of Bounds" where blank is the attribute whose data range constraint was violated.
  - The error message explanation (upon which violation does it take place): If a value that is inserted into a table violates the corresponding data range constraint it will activate a trigger that will output an error message.
  - The error message example according to user(s) scenario(s): If a user inserts age = 4000 into the Tourist table, the error message "Age is out of bounds" will appear and the value will not be inserted into the table.

  Please repeat that pattern for each SQL query.
- The header of the views created in order to facilitate data accesses: Please insert the header of the views as follows:
  - The view created:

```
mysql> Create View `TourG` AS
    -> Select r.rating, t.nameq, e.salary, ts.times, ts.dates
    -> From TourGuide t
    -> Left Join Employee e
    -> ON t.ssn = e.ssn
    -> Left Join Rates r
    -> ON t.ssn = r.otherSSN
    -> Left Join Runs rs
    -> ON t.ssn = rs.ssn
    -> Left Join Tour ts
    -> ON rs.tourid = ts.tourid;
Query OK, 0 rows affected (0.19 sec)

mysql> Select * From TourG;
+--------+---------+--------+-------+---------+
| rating | nameq   | salary | times | dates   |
+--------+---------+--------+-------+---------+
|   NULL | Kristin |  15000 |  1030 | 3242015 |
|      5 | James   |  16000 |  1130 | 3252015 |
|      8 | Opal    |  12000 |  1230 | 3252015 |
|      7 | Opal    |  12000 |  1230 | 3252015 |
+--------+---------+--------+-------+---------+
4 rows in set (0.00 sec)
```

  - The view justification: This is the view for a user that is a tour guide at the park. He or she can only see the ratings of all the tour guides, their names, salaries, and the times and dates of the tours that they run. This allows them to access the parts of the database that are relevant to their job and protect the privacy of other users, like restricting that no one can the social securities numbers of any person at the park.
  - The tables taking part in the view and the specific attributes taking part in the view:
    - The table name: The table name is TourGuideView
    - The table header (all attributes): The attributes are rating, name, salary, times and dates.
  - The trigger built upon those views (if any): Please insert the triggers as follows.
    - The trigger: Please insert the trigger in here.
    - The trigger justification (explanation, correlating it to user scenario(s)): Please insert the justification in here.

    Please repeat for every trigger existing in the view.
- The header of the views created in order to facilitate data accesses: Please insert the header of the views as follows:
  - The view created:

```
mysql> Create View `Vets` AS
    -> Select v.nameq, e.salary, d.species, d.healthStatus, c.dateq, f.type, f.a
mount
    -> FROM Veterinarian v
    -> Left Join Employee e
    -> ON v.ssn = e.ssn
    -> Left Join ChecksOn c
    -> ON c.ssn = v.ssn
    -> Left Join Dinosaur d
    -> ON d.did = c.did
    -> Left Join Food f
    -> ON f.did = d.did;
Query OK, 0 rows affected (0.06 sec)

mysql> SELECT * FROM vets;
ERROR 1146 (42S02): Table 'final.vets' doesn't exist
mysql> SELECT * FROM Vets
    -> ;
+---------+--------+--------------+--------------+---------+-------+--------+
| nameq   | salary | species      | healthStatus | dateq   | type  | amount |
+---------+--------+--------------+--------------+---------+-------+--------+
| Cameron |  30000 | brontosaurus | fair         | 3242014 | hay   |     70 |
| Sara    |  34000 | trex         | good         | 4252015 | sheep |      7 |
| Sean    |  35000 | raptor       | fair         | 6262015 | pigs  |      2 |
+---------+--------+--------------+--------------+---------+-------+--------+
3 rows in set (0.00 sec)
```

  - The view justification: Veterinarians can see the names of other veterinarians, their salaries, the species that they treat, the healthStatus on the day that they were checked on and the amount of food and what type of food can be seen in this view.
    * The table name: The name of the table is vets.
    * The table header (all attributes): The attributes in this table are name, salary, species, healthStatus, date, type and amount.

- The header of the views created in order to facilitate data accesses: Please insert the header of the views as follows:
  - The view created:

```
mysql> CREATE View `Exper` AS
    -> SELECT e.nameq, emp.salary, d.species, f.type, f.amount, t.tricks, d.healthStatus, m.dateq
    -> FROM Expert e
    -> Left Join Employee emp
    -> ON e.ssn = emp.ssn
    -> Left Join Trains t
    -> ON t.ssn = e.ssn
    -> Left Join Monitors m
    -> ON m.ssn = e.ssn
    -> Left Join Dinosaur d
    -> ON d.did = t.did
    -> Left Join Food f
    -> ON d.did = f.did;
Query OK, 0 rows affected (0.14 sec)

mysql> Select * FROM `Exper`;
+---------+--------+--------------+-------+--------+--------+--------------+---------+
| nameq   | salary | species      | type  | amount | tricks | healthStatus | dateq   |
+---------+--------+--------------+-------+--------+--------+--------------+---------+
| Sean    |  12000 | NULL         | NULL  |   NULL | NULL   | NULL         | 3282013 |
| Chad    |  15000 | NULL         | NULL  |   NULL | NULL   | NULL         | 4292015 |
| Chris   |  13000 | NULL         | NULL  |   NULL | NULL   | NULL         | 5152015 |
| Heather |  40000 | brontosaurus | hay   |     70 | sit    | fair         |    NULL |
| Desiree |  35000 | trex         | sheep |      7 | stay   | good         |    NULL |
| Beth    |  31000 | raptor       | pigs  |      2 | speak  | fair         |    NULL |
+---------+--------+--------------+-------+--------+--------+--------------+---------+
6 rows in set (0.00 sec)
```

  - The view justification: Please insert the view justification in here.
  - The tables taking part in the view and the specific attributes taking part in the view: The expert can see the names and salaries of other experts, the species they train, the types of food and tricks the dinosaurs they train know, the health status of the dinosaur that they train and the date that the dinosaur was monitored.
    * The table name: The name of the table is expertview.
    * The table header (all attributes): The attributes for the table are name, salary, species, type, amount, tricks, healthStatus, and date.

II. Project Highlights.

- All the related source code should run on the Ilab machines, and should be included with your submission along with the final LaTeX report, as well as the Power Point Presentation.
- The presentation (7 to 8 minutes) should include the following points (The order of the slides is important):
  1) Title: Project Names (authors and affiliations)
  2) Project Goal
  3) Outline of the presentation
  4) Description
  5) A picture is essential. Interface snapshot and users interaction with it
  6) Project Stumbling Blocks
  7) Data collection, Conceptual Diagram, Integrity constraints
  8) Sample of findings
  9) Future Extensions
  10) Acknowledgements
  11) References and Resources used(libraries, languages, web resources)
  12) Demo(3 minutes)

  Please follow the example SuperFinal presentation mock up that is posted on Sakai, for more details.
- On July 11th, submission should have taken place. Also you have to be ready to do the presentation (7 to 8 minutes) as well as the demo to show the functionalities of the project (3 minutes): every group member should attend the demo (and presentation) and will have to specify the parts he/she has worked upon, in order to allow us to grade all students with fairness.
- Thank you, and best of luck!