Project team #10 - Animal Adoption Agency

Team Members: Aaron Spotts, Sean Hopkins, Yosan Emabye, Zander Cowan, Roger Lin

*Team Member Emails:* [*aspotts@charlotte.edu*](mailto:aspotts@charlotte.edu)*,* [*yembaye@charlotte.edu*](mailto:yembaye@charlotte.edu)*,* [*shopki24@charlotte.edu*](mailto:shopki24@charlotte.edu)*,* [*rlin8@charlotte.edu*](mailto:rlin8@charlotte.edu) *mcowan17@charlotte.edu*

*Presentation Template :* [*https://www.canva.com/design/DAGXO\_59zIA/KvlFuxbkX75w2-D5kgnl7A/edit?utm\_content=DAGXO\_59zIA&utm\_campaign=designshare&utm\_medium=link2&utm\_source=sharebutton*](https://www.canva.com/design/DAGXO_59zIA/KvlFuxbkX75w2-D5kgnl7A/edit?utm_content=DAGXO_59zIA&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)

## PROJECT PROPOSAL

**Content, Scope and Objectives**

This project’s purpose is for our group to learn more about the intricacies of pet adoption management to the purpose of creating an efficient pet adoption process. To achieve this we intend to create a database that allows us to reduce the hurdles when attempting to adopt a pet, manage employees and their responsibilities, keep an up to date inventory of pets available for adoption, along with maintaining information about the business end such as ledgers, inventory, and donors.

## PROJECT ENVIRONMENT

MySQL, MYSQL Workbench because it unifies the development process of our database. The MySQL Workbench is also considered industry standard which is an excellent opportunity for us to become familiar with a popular development tool.

## HIGH LEVEL REQUIREMENTS

### Initial user roles

| **User Role** | **Description** |
| --- | --- |
| Manager | Add, remove, and edit employees, along with duties of employees |
| Employee | Enroll new animals and adopters, view information about animals and adopters |
| Adopter | View animal descriptions, sign waivers and contracts |
| Donor | Donate money to the agency |

### Initial user story descriptions

| **Story ID** | **Story description** |
| --- | --- |
| US1 | As a manager I need to be able to hire and fire employees so I can keep the Agency properly staffed. As a manager, I want the ability to keep track of both my employees, adopters, and animals so that I can properly supervise the Agency. |
| US2 | As an employee, I need to be able to record information about the animals we take into the Agency so we can keep track of all the animals brought into our care. As an employee I also need to collect information about potential adopters and keep them in our records so as to ensure that we are adopting our animals to good owners, and so the owners can be contacted in the future if need be. As an employee I also need to be able to update with new information gathered or provided about our animals by adopters so our records are not limited to what we knew about the animal when first gathered. |
| US3 | As an adopter, I want to be able to see the animals that are available at the agency so I can choose between them. As an adopter I also want to be able to fill out forms and contracts so that I can adopt an animal. |
| US4 | As a donor, I want my donation process to be easy and streamlined. |

## HIGH LEVEL CONCEPTUAL DESIGN

Entities:

Animal

Adopter

Employee

Manager

Employee\_Schedule

Scheduled\_Visits

Donor

Ledger

Inventory

Inventory\_Transactions

Relationships:

Adopter Visits Animal

Adopter Adopts Animal

Employee Cares For Animal

Manager Manages Employee

Manager manages Employee\_Schedule

Manager makes inventory\_transaction

Manager updates ledger

Employee makes inventory\_transaction

Inventory\_Transaction updates inventory

Employee approves Scheduled\_visit

Donor donates to Ledger

An Adopter Schedules a Scheduled\_Visit

An Animal is Scheduled for a Scheduled\_Visit

# Sprint 1

## REQUIREMENTS

Refine the user stories that you made in previous sprint. List your updated user stories and any notes you wish to include in decreasing order of priority and highlight the stories chosen for Sprint 1. *There is no need to show your story refinement process - just the list of updated stories suffices.* Use the format shown below.

| **Story ID** | **Story description** |
| --- | --- |
| US1 | As a Manager, I want to assign employees specific roles so that each person has clear responsibilities and can manage animals, adopters, and inventory effectively. |
| US2 | As an Employee, I want to store information about animals, including their species, type, age, and vaccination status so that adopters can choose animals to adopt from. |
| US3 | As an Employee, I want to store information about adopters, including their preferred species and application status so that background checks and communication with potential adopters can be handled smoothly. |
| US4 | As an Employee, I want to schedule visits between adopters and specific animals so that adoption processes can be organized and streamlined. |
| US5 | As an adopter, I need the scheduling process for a visit to be simple, transparent, and easy in order to find a pet that is appropriate for me. |
| US6 | As a Manager, I want to be able to view the ledger to keep track of the agency’s financial transactions such as donations. |
| US7 | As a Manager, I want to be able to view the inventory in order to keep track of products and manage the inventory. |
| US8 | As a donor, I want to be able to send donations to the agency. |

## CONCEPTUAL DESIGN

Include your detailed conceptual design here. Use the format shown below.

Entity: **Person**

Attributes:

person\_id

role [multi-valued]

first\_name

last\_name

address

dob

phone\_number

email

username

password

Entity: **Animal**

Attributes:

animal\_id

name

species

Type  
 Age

vaccinated

Entity: **Adopter**

Attributes:

person\_id [foreign key]

application\_status

preferred\_species [multi-valued]

Entity: **Donor**

Attributes:

person\_id [foreign key]

role

Entity: **Employee**

Attributes:

person\_id [foreign key]

salary

Entity: **Employee\_Schedule**

Attributes:

person\_id [foreign key]

timeslot [multi-valued]

Entity: **Inventory**

Attributes:

product\_name

product\_id

current\_amount

Entity: **Manager**

Attributes:

person\_id [foreign key]

Entity: **Scheduled\_Visits**

Attributes:

animal\_id [foreign key]

Person\_id [foreign key]

time

grace\_period

Entity: **Ledger**

Attributes:

current\_balance

change\_balance

person\_id[foreign key]

Description

Entity: **Inventory\_Transaction**

Attributes:

transaction\_id

person\_id[foreign key]

product\_id

transaction\_amount

description

Relationship: **Person** assigned\_to **Employee\_Schedule**

Cardinality: Many to One

Participation:

* Person has total participation
* Employee\_Schedule has partial participation

Relationship: **Employee** manages **Animal**

Cardinality: Many to One

Participation:

* Employee has total participation
* Animal has partial participation

Relationship: **Adopter** adopts **Animal**

Cardinality: Many to Many

Participation:

* Adopter has partial participation
* Animal has partial participation

Relationship: **Employee** handles **Adopter**

Cardinality: Many to Many

Participation:

* Employee has total participation
* Adopter has partial participation

Relationship: **Donor** donates\_to **Ledger**

Cardinality: One to Many

Participation:

* Donor has partial participation
* Inventory has partial participation

Relationship: **Adopter** schedules **Scheduled\_Visits**

Cardinality: Many to Many

Participation:

* Adopter has partial participation
* Scheduled\_Visits has total participation

## LOGICAL DESIGN

Include your logical design here. Use the format shown below.

Table: **Person**

Columns:

Person\_id [Primary Key]

role [multi-valued]

first\_name

last\_name

address

dob

phone\_number

email

username

password

Table: **Animal**

Columns:

animal\_id[Primary Key]

name

species

Type  
 Age

vaccinated

Table: **Adopter**

Columns:

person\_id [foreign key; references Person\_id of table Person]

application\_status

preferred\_species [multi-valued]

Table: **Donor**

Columns:

person\_id [foreign key; references Person\_id of table Person]

role

Table: **Employee**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Salary

Table: **Employee Schedule**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Timeslot [multivalued]

Table: **Inventory**

Columns:

product\_name

product\_id[Primary Key]

current\_amount

Table: **Manager**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Table: **Scheduled Visits**

Columns:

animal\_id [foreign key; references animal\_id of table Animal]

Person\_id [foreign key; references Person\_id of table Person]

time

grace\_period

Table: **Ledger**

Columns:

current\_balance

change\_balance

person\_id[foreign key; references Person\_id of table Person]

Description

Table: **Inventory\_Transactions**

Columns:

transaction\_id

person\_id[foreign key; references Person\_id in Person table]

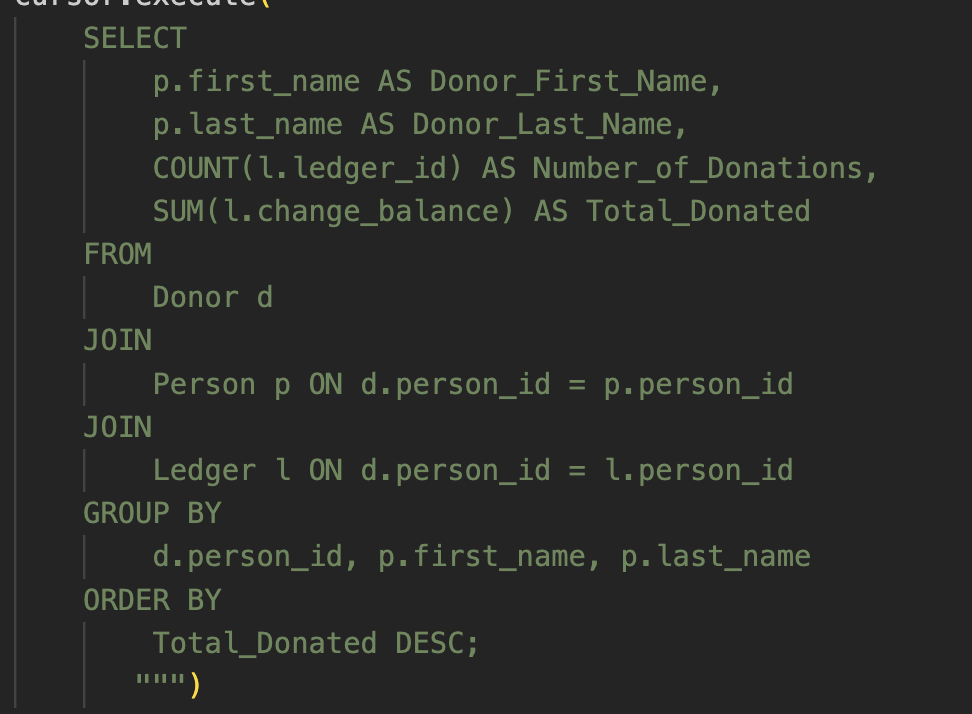
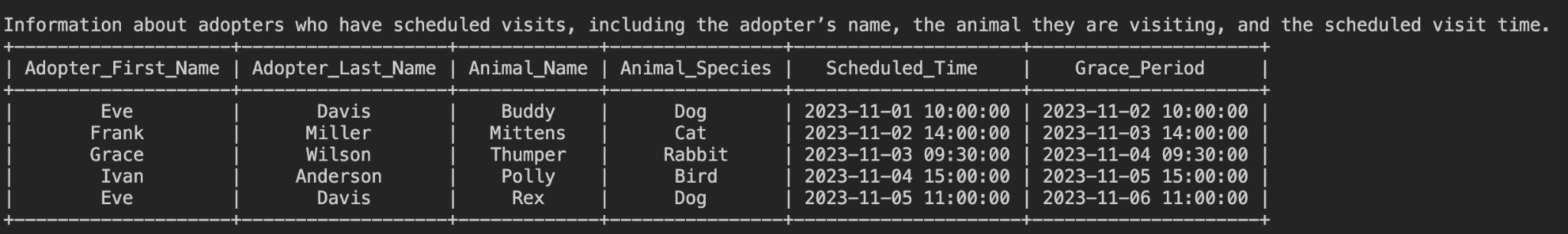
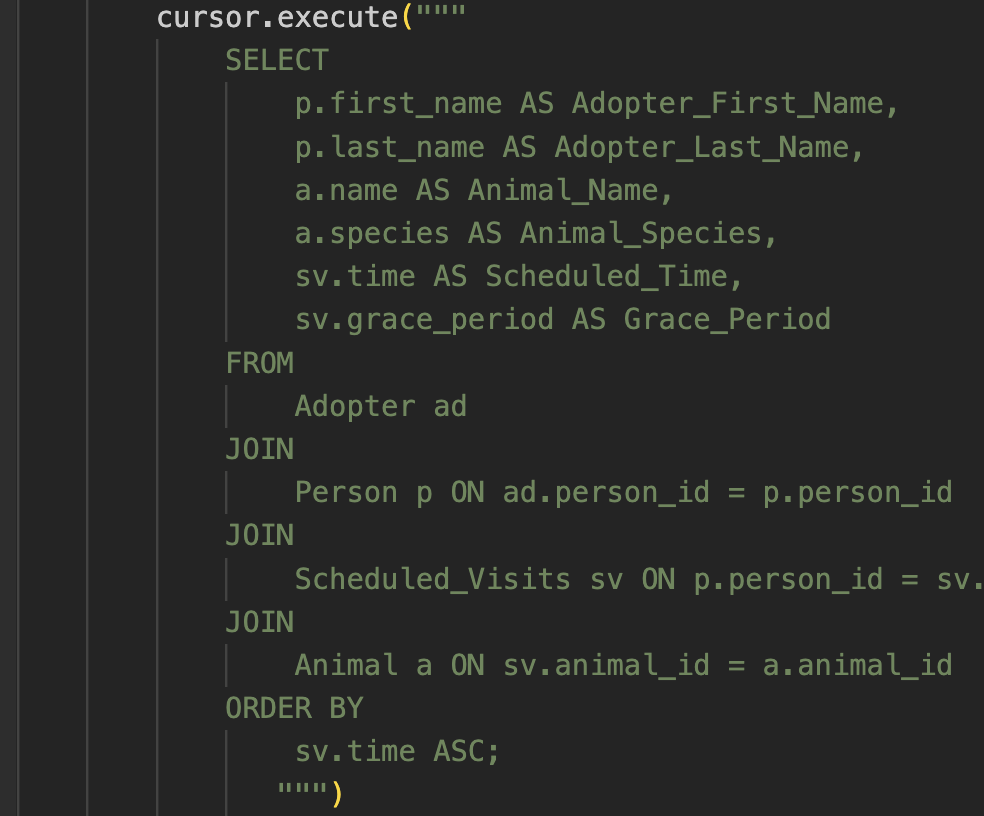
product\_id

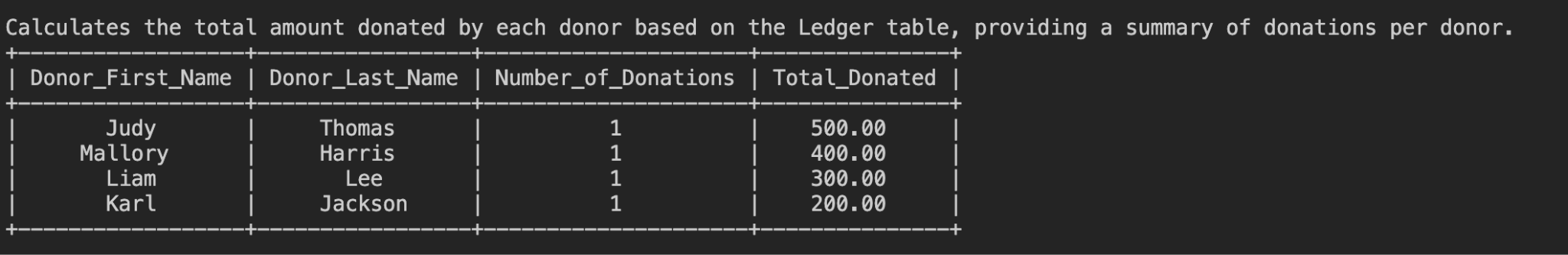
transaction\_amount

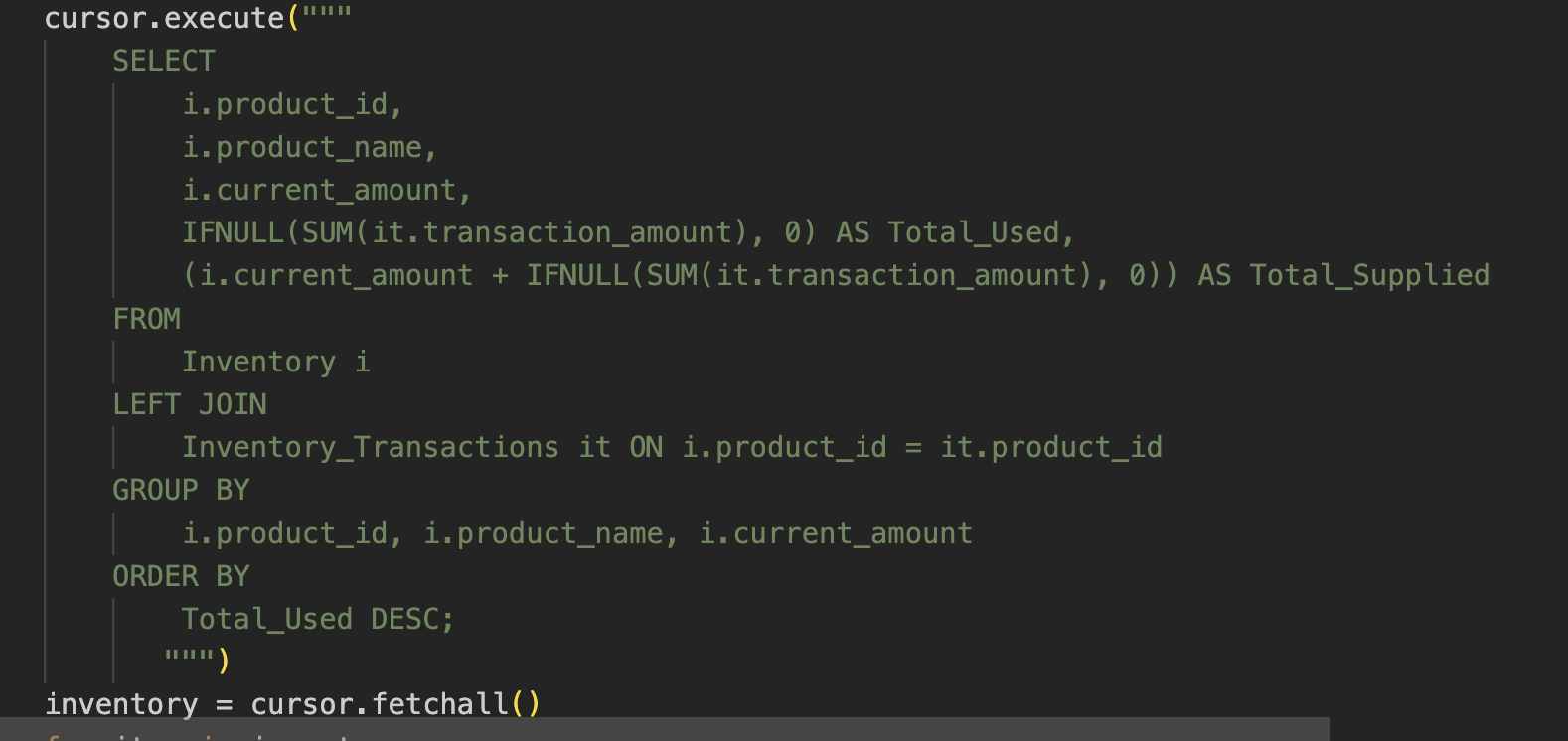
description

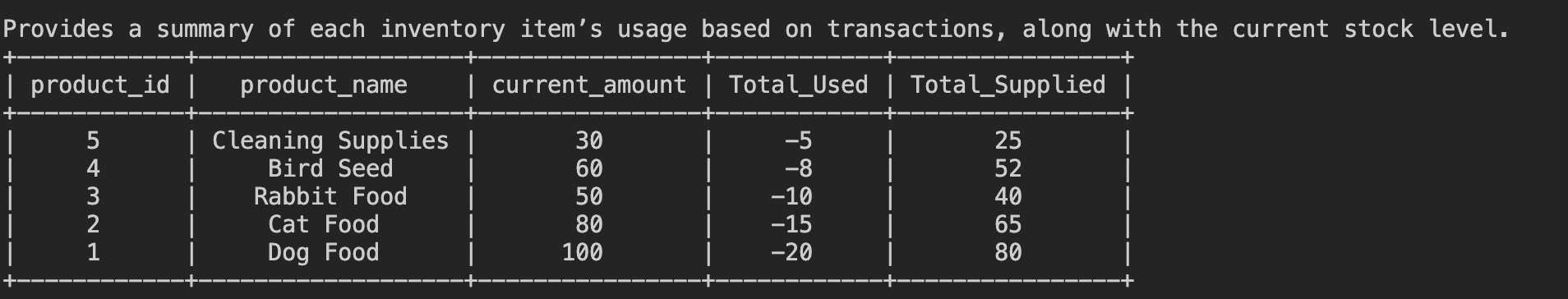
## SQL QUERIES

List at least **three** SQL queries that perform data retrievals relevant to the features chosen in the current sprint. For each query, paste a **screenshot** of the output, as shown through the database management tool.









Sprint 2

## REQUIREMENTS

Refine the user stories that you made in the previous sprint. List your updated user stories in decreasing order of priority. Highlight the stories for which database design was completed in Sprint 1 in one color. Highlight the updated/new stories chosen for Sprint 2 in a different color. *There is no need to explicitly show your story refinement process.* Use the format shown below.

| **Story ID** | **Story description** |
| --- | --- |
| US1 | As an Employee, I want to store information about animals, including species, breed, age, and vaccination status, so that adopters can easily view details about animals available for adoption and make informed choices. |
| US2 | As an Employee, I want to store detailed information about adopters, including their preferred species, application status, and contact details, so that background checks, adoption approvals, and follow-up communication can be managed smoothly and efficiently. |
| US3 | As an Employee, I want to schedule visits between adopters and specific animals based on the adopter's species preference so that the adopter can find the right companion for them. |
| US4 | As an Adopter, I want to be able to view my application status in order to track my application progress. |
| US5 | As a Manager, I want to assign schedules to employees to ensure proper staffing. |
| US6 | As a Manager, I want to view and manage the inventory to keep track of products and to make sure that the agency has enough products in inventory to keep operations running. |
| US7 | As a Manager, I want to see donor transactions, so that I can track and manage inventory effectively. |
| US8 | As a Manager, I want to be able to view the current balance of the ledger to keep track of the agency’s financial transactions and to change the balance of the ledger so that all financial transaction data is up to date. |
| US9 | As a Person I want to create an account that holds all my data for the following processes as either an employee, adopter, or donor |
| US10 | As a Person I want to donate to the agency and view my past donations |

## CONCEPTUAL DESIGN

Include your complete updated conceptual design here. Use the format shown below.

Entity: **Person**

Attributes:

* person\_id [PK]
* first\_name
* last\_name
* address
* dob
* phone\_number
* email [unique]
* role [multi-valued]
* hashed\_password

Entity: **Breed**

Attributes:

* breed [PK]
* species

Entity: **Animal**

Attributes:

* animal\_id [PK]
* Name
* breed[FK references breed in Breed]
* Age
* vaccinated

Entity: **Employee**

Attributes:

* person\_id [PK, FK references person\_id in Person]
* salary

Entity: **Manager**

Attributes:

* person\_id [PK, FK references person\_id in Employee]

Entity: **Adopter**

Attributes:

* person\_id [PK, FK references person\_id in Employee]
* Application\_status
* Preferred\_species

Entity: **Scheduled\_Visits**

Attributes:

* animal\_id [PK, FK references animal\_id in Animal]
* person\_id [PK, FK references person\_id in Person]
* visit\_time [PK]
* grace\_period

Entity: **Employee\_Schedule**

Attributes:

* person\_id [PK, FK references person\_id in Employee]
* start\_time [PK]
* end\_time

Entity: **Inventory**

Attributes:

* product\_id [PK]
* Product\_name [UNIQUE]
* current\_amount

Entity: **Inventory\_Transactions**

Attributes:

* transaction\_id [PK]
* person\_id [FK references person\_id in Person]
* product\_id [FK references product\_id in Inventory]
* transaction\_amount
* Description
* transaction\_time

Entity: **Ledger**

Attributes:

* edger\_id [PK]
* person\_id [FK references person\_id in Person]
* change\_amount
* description
* Description
* Transaction\_time

Entity: **Medical\_Records**

Attributes:

* record\_id [PK]
* animal\_id [FK references animal\_id in ANimal]
* ehr\_system
* ehr\_record\_id
* Description
* transaction\_time

Relationship: **Employee** is\_a **Person**

Cardinality: One-to-One

Participation:

* Employee has total participation
* Person has partial participation

Relationship: **Adopter** is\_a **Person**

Cardinality: One-to-One

Participation:

* Adopter has total participation
* Person has partial participation

Relationship: **Adopter** adopts **Animal**

Cardinality: Many to One

Participation:

* Adopter has partial participation
* Animal has partial participation

Relationship: **Person** donates\_to **Ledger**

Cardinality: Many to One

Participation:

* Donor has partial participation
* Ledger has partial participation

Relationship: **Employee** updates **Ledger**

Cardinality: One to One

Participation:

* Employee has partial participation
* Ledger has partial participation

Relationship: **Adopter** schedules **Scheduled\_Visits**

Cardinality: Many to Many

Participation:

* Adopter has partial participation
* Scheduled\_Visits has total participation

Relationship: **Animal** is visited during **Scheduled\_Visits**

Cardinality: Many to Many

Participation:

* Animal has partial participation
* Scheduled\_Visits has total participation

Relationship: **Animal** has **Medical\_Records**

Cardinality: One to One

Participation:

* Animal has partial participation
* Medical\_Records has total participation

Relationship: **Animal** has a **Breed**

Cardinality: One to One

Participation:

* Animal has partial participation
* Breed has total participation

Relationship: **Employee** creates **Inventory\_transaction**

Cardinality: One to Many

Participation:

* Employee has partial participation
* Inventory\_transaction has total participation

Relationship: **Inventory\_transaction** updates **Inventory**

Cardinality: One to Many

Participation:

* Inventory\_transaction has total participation
* Inventory has total participation

## LOGICAL DESIGN WITH NORMAL FORM IDENTIFICATION

Table: **Person**

Columns:

person\_id [PK]

first\_name

last\_name

address

dob

phone\_number

Email

Hashed\_password

Highest normalization level: BCNF

Table: **Animal**

Columns:

animal\_id [PK]

name

breed [fk]

age

vaccinated

Highest normalization level: BCNF

Table: Medical Records

Columns:

record\_id [PK]

animal\_id [fk]

ehr\_record\_id

ehr\_system   
 last\_updated

Table: **Breed**

Columns:

breed [PK]

species

Highest normalization level: BCNF

Table: **Adopter**

Columns:

person\_id [PK/FK] [references person\_id of Person]

application\_status

Preferred\_species

Justification of primary key: person\_id uniquely identifies each adopter, which links to a unique person in the Person table.

Highest normalization level: BCNF

Table: **Employee**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Salary  
  
Highest normalization level: BCNF

Table: **Employee Schedule**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Timeslot [multivalued]  
  
Highest normalization level: BCNF

Justification of primary key: The composite primary key (person\_id, timeslot) ensures that each schedule entry is unique to an employee and a specific time slot.

Table: **Inventory**

Columns:

product\_id [PK]

product\_name

current\_amount

Highest normalization level: BCNF

Table: **Manager**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Highest normalization level: BCNF

Table: **Scheduled Visits**

Columns:

animal\_id [foreign key; references animal\_id of table Animal]

Person\_id [foreign key; references Person\_id of table Person]

time

Grace\_period  
Highest normalization level: BCNF

Table: **Ledger**

Columns:

Ledger\_id [PK]

current\_balance

change\_balance

person\_id[foreign key; references Person\_id of table Person]

Description  
  
Highest normalization level: BCNF

Table: **Inventory\_Transactions**

Columns:

transaction\_id

person\_id[foreign key; references Person\_id in Person table]

product\_id [FK references product\_id in inventory]

transaction\_amount

Description

Highest normalization level: BCNF

Table: **Medical\_Records**

Columns:

record\_id [PK]

animal\_id [FK; references `Animal`(`animal\_id`)]

ehr\_record\_id

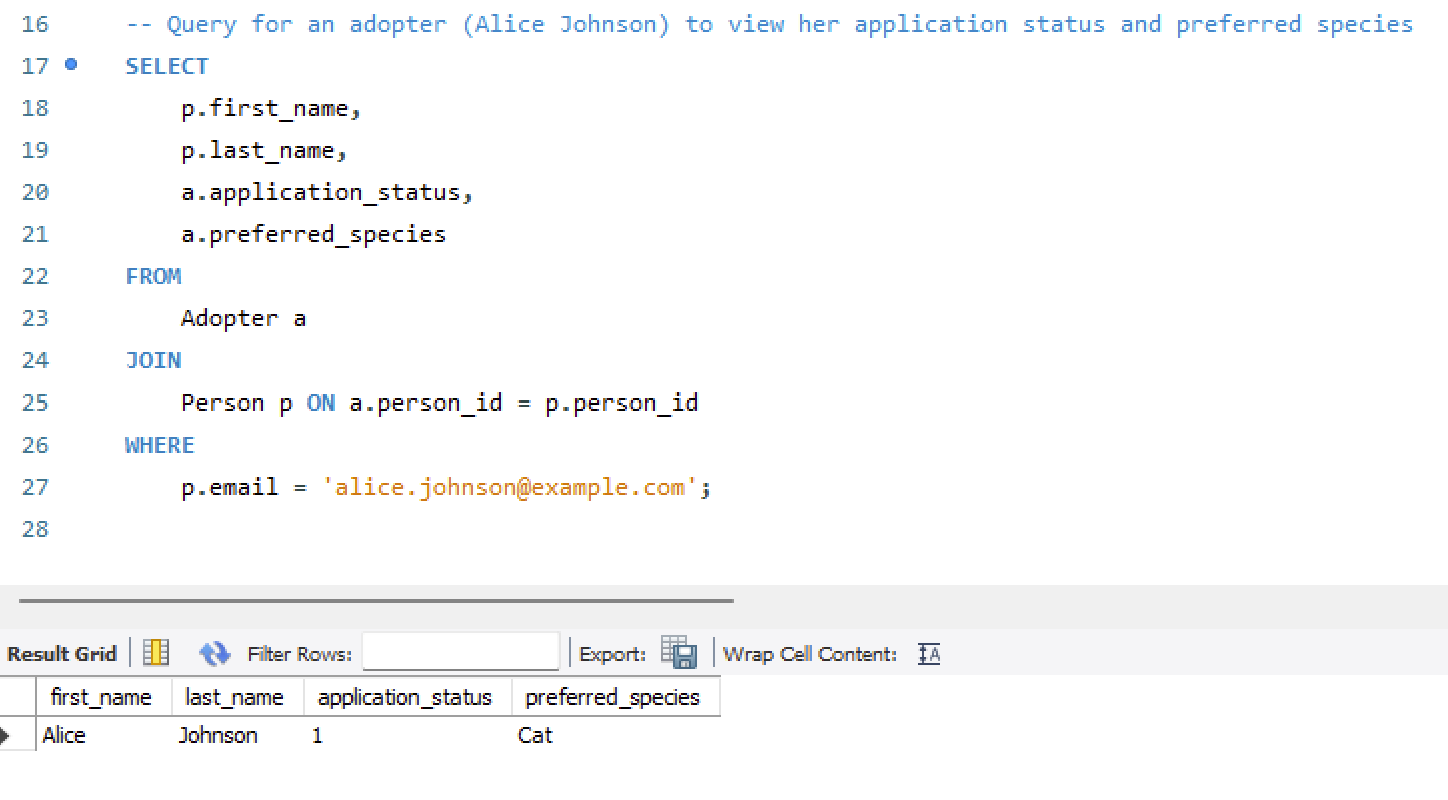
ehr\_system

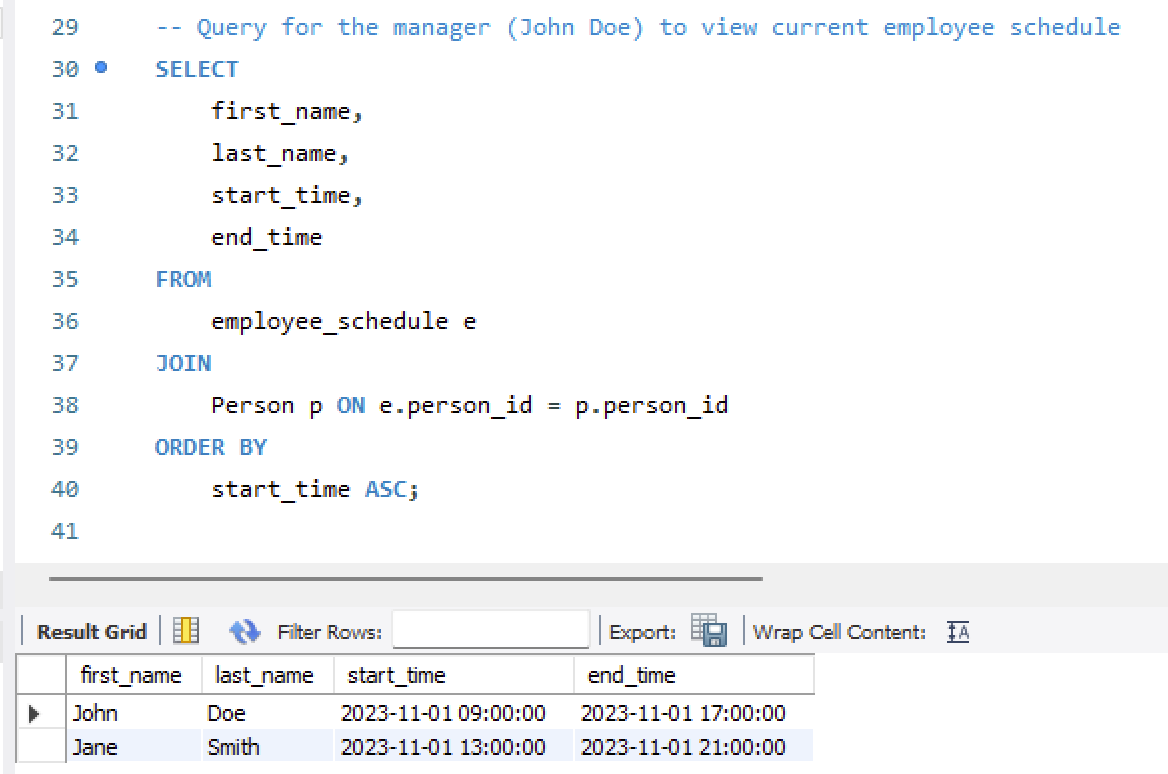
Last\_updated

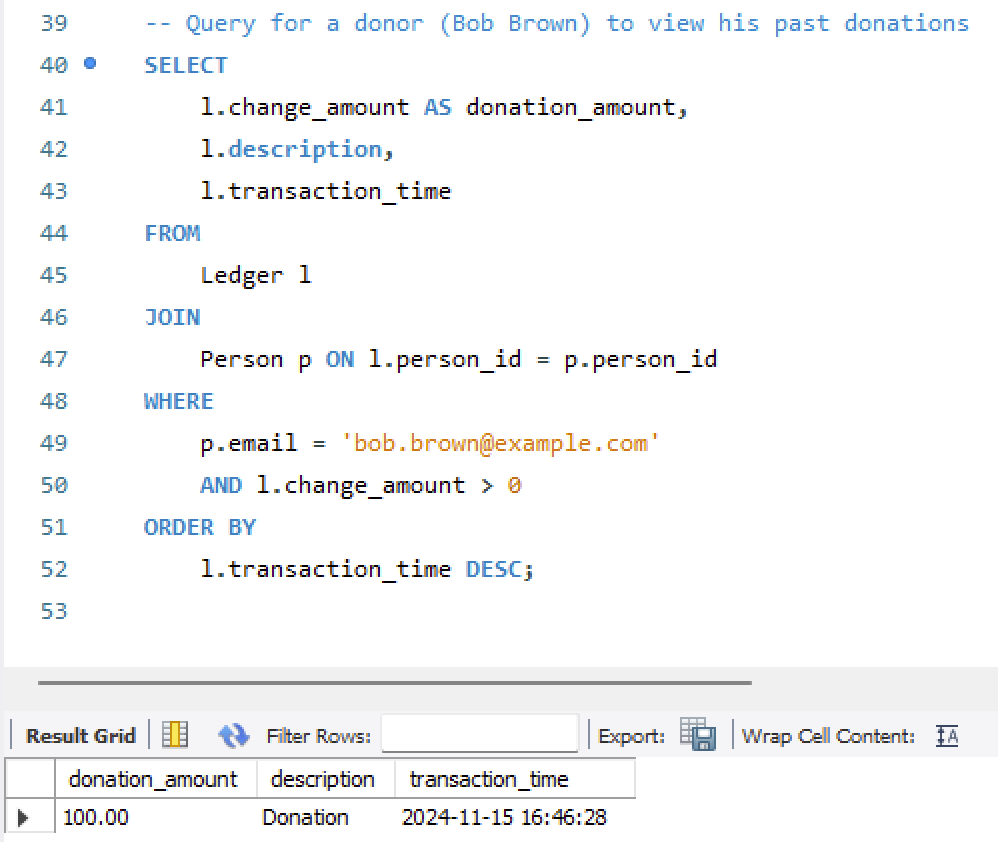
Highest normalization level: BCNF

## SQL QUERIES

Refine your SQL queries that you designed in the previous sprint if in need. List at least **three** SQL queries that perform data retrievals relevant to the features chosen in the current sprint. For each query, paste a **screenshot** of the output, as shown through your user interface.







Sprint 3

## REQUIREMENTS

Refine the user stories that you made in previous sprint. List your updated user stories in decreasing order of priority. Highlight the stories that were completed in Sprint 1 in one color. Highlight the stories that were completed in Sprint 2 in a different color. Highlight the updated/new stories chosen for Sprint 3, if any, in a third color. *There is no need to explicitly show your story refinement process.* Use the format shown below.

| **Story ID** | **Story description** |
| --- | --- |
| US1 | As a new user(either an employee, adopter, or just a person) I want to be able to create a new account |
| US2 | As an Employee, I want to store and view information about animals, including species, breed, age, and vaccination status, so that adopters can easily view details about animals available for adoption and make informed choices. |
| US3 | As an Adopter I want to easily view details about animals available for adoption and make informed choices. |
| US4 | As an Employee, I want to store and view detailed information about user accounts, including their preferred species, application status, and contact details, so that background checks, adoption approvals, and follow-up communication can be managed smoothly and efficiently. |
| US5 | As an Employee, I want to schedule visits between adopters and specific animals based on the adopter's species preference so that the adopter can find the right companion for them. |
| US6 | As an Adopter, I want to be able to view my application status in order to track my application progress. |
| US7 | As a Manager, I want to assign schedules to employees to ensure proper staffing. |
| US8 | As a Manager, I want to view and manage the inventory to keep track of products and to make sure that the agency has enough products in inventory to keep operations running. |
| US9 | As a Manager, I want to see donor transactions, so that I can track and manage the budget effectively. |
| US10 | As a Manager, I want to be able to view the current balance of the ledger to keep track of the agency’s financial transactions and to change the balance of the ledger so that all financial transaction data is up to date. |
| US11 | As a Person I want to donate to the agency and view my past donations |

## CONCEPTUAL DESIGN

Include your complete updated conceptual design here. Use the format shown below.

Entity: **Person**

Attributes:

* person\_id [PK]
* first\_name
* last\_name
* address
* dob
* phone\_number
* email [unique]
* role [multi-valued]
* password\_hash

Entity: **Breed**

Attributes:

* breed [PK]
* species

Entity: **Animal**

Attributes:

* animal\_id [PK]
* name
* breed[FK references breed in Breed]
* age
* vaccinated

Entity: **Employee**

Attributes:

* person\_id [PK, FK references person\_id in Person]
* salary
* role
* start\_date

Entity: **Manager**

Attributes:

* person\_id [PK, FK references person\_id in Employee]

Entity: **Adopter**

Attributes:

* person\_id [PK, FK references person\_id in Employee]
* Application\_status
* Preferred\_species

Entity: **Scheduled\_Visits**

Attributes:

* animal\_id [PK, FK references animal\_id in Animal]
* person\_id [PK, FK references person\_id in Person]
* visit\_time [PK]
* grace\_period

Entity: **Employee\_Schedule**

Attributes:

* person\_id [PK, FK references person\_id in Employee]
* start\_time [PK]
* shift\_length

Entity: **Inventory**

Attributes:

* product\_id [PK]
* Product\_name [UNIQUE]
* current\_amount

Entity: **Inventory\_Transactions**

Attributes:

* transaction\_id [PK]
* person\_id [FK references person\_id in Person]
* product\_id [FK references product\_id in Inventory]
* transaction\_amount
* Description
* Transaction\_time

Entity: **Ledger**

Attributes:

* ledger\_id [PK]
* person\_id [FK references person\_id in Person]
* change\_amount
* description
* Transaction\_time
* account\_id[FK References account\_id in accounts]

Entity: **Accounts**

Attributes:

* Account\_id[PK]
* Current\_balance

Entity: **Medical\_Records**

Attributes:

* record\_id [PK]
* animal\_id [FK references animal\_id in ANimal]
* ehr\_systemx
* ehr\_record\_id
* Description
* transaction\_time

Relationship: **Employee** is\_a **Person**

Cardinality: One-to-One

Participation:

* Employee has total participation
* Person has partial participation

Relationship: **Adopter** is\_a **Person**

Cardinality: One-to-One

Participation:

* Adopter has total participation
* Person has partial participation

Relationship: **Adopter** adopts **Animal**

Cardinality: Many to One

Participation:

* Adopter has partial participation
* Animal has partial participation

Relationship: **Person** donates\_to **Ledger**

Cardinality: Many to One

Participation:

* Donor has partial participation
* Ledger has partial participation

Relationship: **Employee** updates **Ledger**

Cardinality: One to One

Participation:

* Employee has partial participation
* Ledger has partial participation

Relationship: **Adopter** schedules **Scheduled\_Visits**

Cardinality: Many to Many

Participation:

* Adopter has partial participation
* Scheduled\_Visits has total participation

Relationship: **Animal** is visited during **Scheduled\_Visits**

Cardinality: Many to Many

Participation:

* Animal has partial participation
* Scheduled\_Visits has total participation

Relationship: **Animal** has **Medical\_Records**

Cardinality: One to One

Participation:

* Animal has partial participation
* Medical\_Records has total participation

Relationship: **Animal** has a **Breed**

Cardinality: One to One

Participation:

* Animal has partial participation
* Breed has total participation

Relationship: **Employee** creates **Inventory\_transaction**

Cardinality: One to Many

Participation:

* Employee has partial participation
* Inventory\_transaction has total participation

Relationship: **Inventory\_transaction** updates **Inventory**

Cardinality: One to Many

Participation:

* Inventory\_transaction has total participation
* Inventory has total participation

## LOGICAL DESIGN WITH HIGHEST NORMAL FOR\

## MS AND INDEXES

Include your complete updated logical design here. Use the format shown below.

Table: **Table1**

Columns:

pk\_1

column\_1a

column\_1b

*Justification of primary key (if needed)*

Highest normalization level: <1NF/2NF/3NF/BCNF>

Justification (if below BCNF):

Indexes:

Index #: <type (clustered/non-clustered)>

Columns: <ordered list of columns forming the index>

Justification:

Table: **Person**

Columns:

person\_id [PK]

first\_name

last\_name

address

dob

phone\_number

email

password\_hash

Highest normalization level: BCNF

Indexes:   
 Email: non clustered (makes searching for a specific person’s user account by email faster)

Last\_name: non clustered(makes searching for a specific person’s user account by last name faster)

Table: **Animal**

Columns:

animal\_id [PK]

name

breed [FK]

age

vaccinated

Highest normalization level: BCNF  
  
 Indexes:  
 breed: non clustered(speeds up finding specific animals based on breed)

Table: **Breed**

Columns:

breed [PK]

species

Highest normalization level: BCNF

Indexes:  
 species: clustered (speeds up filtering breed by species)

Table: **Adopter**

Columns:

person\_id [PK/FK] [references person\_id of Person]

application\_status

Preferred\_species

Justification of primary key: person\_id uniquely identifies each adopter, which links to a unique person in the Person table.

Highest normalization level: BCNF

Indexes:  
 application\_status: non clustered (speeds up queries based by application status)

Table: **Employee**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Salary

Start\_date

role  
  
Highest normalization level: BCNF

Indexes:  
 salary: non clustered (makes filtered by salary ranges faster)

Role: non clustered(provides a quick lookup of an employees role for the privileges available to their user account)

Table: **Employee Schedule**

Columns:

person\_id [foreign key; references Person\_id of table Person]

Start\_time [multivalued]

shift\_length  
  
Highest normalization level: BCNF

Justification of primary key: The composite primary key (person\_id, timeslot) ensures that each schedule entry is unique to an employee and a specific time slot.  
  
Indexes:  
 person\_id, start\_time: non clustered (speed up queries for employee schedule)

Table: **Inventory**

Columns:

product\_id [PK]

product\_name

current\_amount

Highest normalization level: BCNF

Indexes:  
 Product name: non clustered (speed up for searching by product name)

Table: **Scheduled Visits**

Columns:

animal\_id [foreign key; references animal\_id of table Animal]

Person\_id [foreign key; references Person\_id of table Person]

visit\_time

Grace\_period  
Highest normalization level: BCNF  
  
Indexes:  
 person\_id, visit\_time: non clustered (speed up for scheduled visits by person and time)

Table: **Ledger**

Columns:

ledger\_id [PK]

change\_balance

person\_id[FK; references Person\_id of table Person]

description

Transaction\_time

account\_id[FK; references account\_id of accounts table]  
  
Highest normalization level: BCNF

Indexes: person\_id: non-clustered(speeds up searching the changes made by each person)

Account\_id: non-clustered(speeds up searching the changes made to each account)

Table: **Accounts**

Columns:

Account\_id[PK]

current\_balance

Highest normalization level: BCNF

Indexes: Account\_id: clustered(speeds up checking for a specific account)

Table: **Inventory\_Transactions**

Columns:

transaction\_id

person\_id[foreign key; references Person\_id in Person table]

product\_id [FK references product\_id in inventory]

transaction\_amount

Description

transaction\_time

Highest normalization level: BCNF

Indexes:   
 product\_id: non-clustered(speeds up searching for transactions by product)  
 Person\_id: non-clustered (speeds up transactions based by person)

Table: **Medical\_Records**

Columns:

record\_id [PK]

animal\_id [FK; references `Animal`(`animal\_id`)]

ehr\_record\_id

ehr\_system

Last\_updated

Highest normalization level: BCNF

Indexes:   
 animal\_id: non-clustered(speeds up searching for medical records by animal)

## VIEWS AND STORED PROGRAMS

List the views relevant to your application here. Use the format specified below.

**View**: <name of view>

Goal: <1-2 sentence description of what the view contains and what its purpose is (e.g., why and what user(s) would use it, etc.)>

CREATE VIEW weeklySchedule AS

SELECT first\_name, last\_name, start\_time, shift\_length FROM employeeSchedule e JOIN Person p

ON e.person\_id = p.person\_id;

Goal: The view contains information about employees and their work schedule. This view would be useful for the manager to view the schedule of an employee or employees and in ensuring proper staffing at all times.

CREATE VIEW mostRecentLedgerChange AS

SELECT \*

FROM Ledger

WHERE ledger\_id = (SELECT MAX(ledger\_id) FROM Ledger);

Goal: The view contains the current\_balance and change\_balance of the most recent ledger, allowing managers to quickly check the most recent expenditure to make sure its a legitimate one.

CREATE VIEW adopterApplications AS

SELECT person\_id, application\_status, preferred\_species

FROM Adopter;

Goal: This view contains the application status, preferred species, and id of adopters, allowing employees and managers to quickly view adopters' applications to manage approvals.

CREATE VIEW availableAnimals AS

SELECT a.animal\_id, a.name, a.breed, b.species, a.age, a.vaccinated

FROM Animal a

JOIN Breed b ON a.breed = b.breed

WHERE NOT EXISTS (

SELECT 1 FROM Scheduled\_Visits sv

WHERE sv.animal\_id = a.animal\_id

);

Goal : The view contains the animal id, name, breed, age , and vaccinated status, which are basic information for an animal. It also excludes animals who are currently scheduled for a visit. The view would be useful for adopters who would like to view available animals and their basic info.

CREATE VIEW upcoming\_visitations AS  
SELECT visit\_time, first\_name, last\_name, name

FROM scheduled\_visits s JOIN animal a ON s.animal\_id=a.animal\_id

JOIN person p ON s.person\_id = p.person\_id

ORDER BY visit\_time ASC;

Goal: This view contains the first name, last name, and the visit time of adopters along with the animal they are visiting. An employee would use this view to see a list of upcoming visits to prepare the animal ahead of time.

CREATE VIEW LowInventory AS

SELECT product\_id, product\_name, current\_amount

FROM Inventory

WHERE current\_amount <= 20;

Goal: This view contains the product id, name, and amount from inventory that fall beneath a predetermined threshold. This can be used in an alert system to notify an manager ahead of time of the products that need to be restocked before they run out.

CREATE VIEW DonationsList AS

SELECT change\_amount, description, transaction\_time

FROM Ledger l JOIN Person p ON l.Person\_ID = p.Person\_ID

WHERE change\_amount > 0

ORDER BY transaction\_time DESC;

Goal: This view allows managers to make a list of donations received by the Agency.

List the stored programs relevant to your application thus far here. Use the format specified below for the different kinds of stored programs. Note: if you do not have a particular type of stored program in your application, just leave that part out.

**Stored procedure**: UpdateAccountBalance

Parameters: p\_id IN, c\_amount IN, descr IN, acc\_id IN, current\_balance OUT

Goal: Keep an up-to-date record of how much money the agency currency has

**Stored procedure**: vaccinate\_animal

Parameters: ani\_id IN vaccinated OUT

Goal: Marks an identified animal as a vaccinated animal.

