

# **poughkeepsie**



# **ZOO**

# Table Of Contents

---

- High Level Design**.....3
- Low Level Design**.....4
- States Table.....5
- Employees Table.....6
- Managers .....8
- Cashiers .....9
- Security .....10
- Janitors .....11
- Caretakers .....12
- Directors .....13
- Tour Guides .....14
- Species .....15
- Environments .....16
- Animals .....17
- Security Duties .....19
- Shows .....20
- Show Cast .....21
- Tour Details .....22
- Tours .....23
- Janitor Duties .....24
- Reports** .....25
- Student Reimbursement for Cashiers .....25
- Tours Per Day .....26
- Number of Animals Per Caretaker .....27
- Views** .....28
- Show Director View .....28
- Caretaker View .....30
- Stored Procedures** .....31
- Weather Resistant .....31
- Security** .....32
- Problems** .....34

# High Level Design

---

This section provides an overview of the Poughkeepsie Zoo, the short and long term goals, as well as the implementation of its database systems project.

---

## Overview

The Poughkeepsie Zoo is a entertainment and recreational facility new to the Hudson Valley area. The zoo hopes to bring in families from all around to see their fabulous exhibits. With this installation of this new high tech animal environment, the owner of the facility would like its database systems as new as the zoo itself. The database itself will contain information on employees, animals, and attractions.

The Poughkeepsie Zoo has plans to include on its opening day:

- 7 different environments
- over 50 different kinds of animals
- 6 different entertainment attractions
- 9 different ways to experience the environments and animals
- a staff of more than 40 people

The owner would like the system to assist in the expansion and growth of the zoo.

---

## Short-Term Goals

The short-term goals of implementing this database system would be to facilitate the running of the zoo by keeping electronic records on employees, animals, and attractions.

---

## Long-Term Goals

The long-term goals of implementing this database system would be to facilitate the expansion of the zoo.

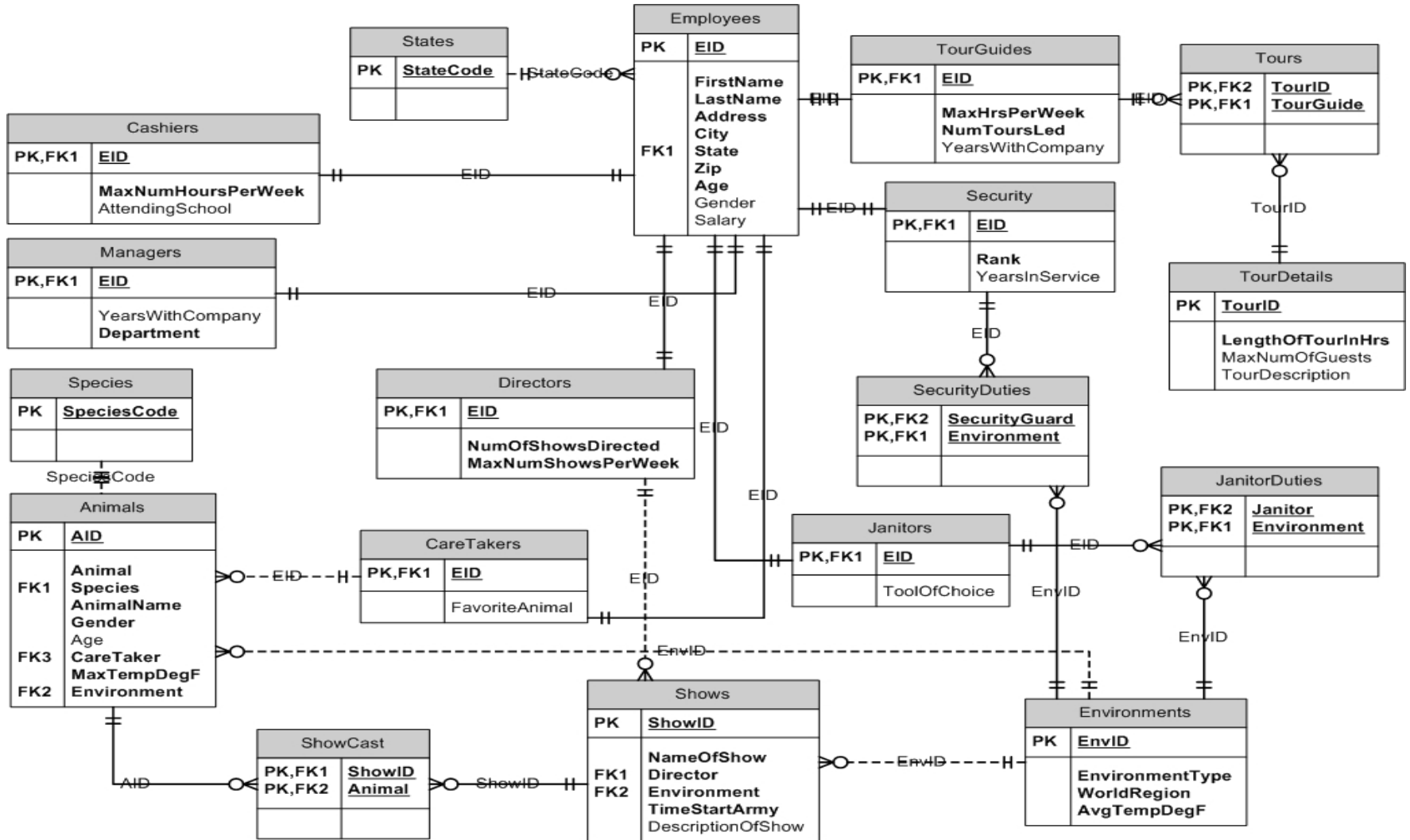
---

## Implementation

The Poughkeepsie Zoo database will be implemented as a set of normalized entities that are somehow related to each other. On first implementation the database will include employees, animal, environment, show, tour, security, and janitorial tables.

# Low Level Design

This section provides the specifics of the Poughkeepsie Zoo. Each section is an entity inside of the database which contains the attributes that describe that entity. Each individual section will include the SQL create statements, a brief description of the entity's purpose, the functional dependencies, and some sample data. The following Entity Relationship Diagram shows how each entity will be working together at a bird's eye view of the project.



# States Table

---

This table will contain a list of all valid state codes. The primary key is StateCode which will stand for the 2 letter code of that particular state.

---

## Functional Dependencies

StateCode →

---

## SQL Create Statement

```
create table states (  
    StateCode char(2) not null,  
    primary key (StateCode)  
)
```

---

## Sample Data

StateCode
AR
AZ
CA
CT
FL
MI
MS
NJ
NY
SC
WY

# Employees Table

---

This table will contain a list of all employees. The primary key is EID which stands for Employee ID.

---

## Functional Dependencies

EID → FirstName, LastName, Address, City, State, Zip, Age, Gender, SalaryUSD

---

## SQL Create Statement

```
create table employees (  
    EID char(10) not null,  
    FirstName varchar(50) not null,  
    LastName varchar(50) not null,  
    Address varchar(100) not null,  
    City varchar(50) not null,  
    State char(2) not null,  
    Zip char(5) not null,  
    Age int not null,  
    Gender char(1) check(Gender='M' or Gender='F'),  
    SalaryUSD decimal(12,2),  
    primary key(EID),  
    foreign key(State) references states(StateCode)  
)
```

---

## Sample Data

EID	FirstName	LastName	Address	City	State	Zip	Age	Gender	SalaryUSD
e00001	Steve	Dave	24 Mallrats Ln	Trenton	NJ	07809	24	M	30000.00
e00002	Amy	Gotti	427 Hill Dr	Bohemia	NY	11681	20	F	15000.00
e00005	Jim	Weir	900 Web Way	Poughkeepsie	NY	12603	30	M	200000.00
e00006	Muffin	Man	333 Drury Ln	Bakersville	NC	16549	70	M	10000.00
e00008	Alan	Labouseur	3 NF Rd	Hyde Park	NY	12453	29	M	200000.00
e00016	Michael	Myers	13 Halloween Dr	Cleveland	OH	36962	34	M	91000.00
e00017	Andy	Dick	631 Main St	Charleston	SC	67922	41	M	75000.00
e00018	John	Stamos	654 Full House Rd	Riverside	CA	98524	39	M	125000.00
e00025	Austin	Powers	69 Shagadelic Dr	Mountainville	MN	68451	38	M	175000.00
e00026	James	Bond	007 Agent Dr	San Francisco	CA	94132	42	M	58000.00
e00032	Obi Wan	Kenobi	577 Far Far Away	Tatooine	OR	78412	67	M	100000.00
e00035	Oscar	Grouch	9874 Garbage Pail Pl	Sesame	TX	52010	48	M	90000.00
e00036	Curtis	Martin	24 Runner Rd	Little Falls	NJ	07606	35	M	175000.00

# Managers Table

---

This table will contain a list of all the employees that are managers and the data that pertains to them. The primary key is EID which stands for Employee ID.

---

## Functional Dependencies

EID → YearsWithCompany, Department

---

## SQL Create Statement

```
create table managers (  
    EID char(10) not null,  
    YearsWithCompany int,  
    Department varchar(50) not null,  
    primary key(EID),  
    foreign key(EID) references employees(EID)  
)
```

---

## Sample Data

EID	YearsWithCompany	Department
e00005	10	Cashier Manager
e00008	15	Caretaker Manager
e00012	7	Tour and Show Manager
e00032	20	Security and Janitor Manager
e00041	10	Zoo Manager

# Cashiers Table

---

This table will contain a list of the employees that are cashiers and the data pertaining to them. The primary key is EID which stands for Employee ID.

---

## Functional Dependencies

EID → MaxNumHrsPerWeek, AttendingSchool

---

## SQL Create Statement

```
create table cashiers (  
    EID char(10) not null,  
    MaxNumHrsPerWeek int not null,  
    AttendingSchool char(1) default 'N' check(AttendingSchool='Y' or AttendingSchool ='N'),  
    primary key(EID),  
    foreign key(EID) references employees(EID)  
)
```

---

## Sample Data

EID	MaxNumHrsPerWeek	AttendingSchool
e00002	40	Y
e00005	40	N
e00009	15	N
e00014	30	N
e00020	25	Y
e00024	25	Y
e00038	35	N

# Security Table

---

This table will contain a list of the members of the security force and the data that describes them. The primary key is EID which stands for Employee ID.

---

## Functional Dependencies

EID → Rank, YearsInService

---

## SQL Create Statement

```
create table security (  
    EID                char(10)    not null,  
    Rank               varchar(30) not null,  
    YearsInService     int,  
    primary key(EID),  
    foreign key(EID) references employees(EID)  
)
```

---

## Sample Data

EID	Rank	YearsInService
e00007	Policeman	5
e00022	Detective	15
e00025	Secret Agent	25
e00026	Secret Agent	30
e00028	Enforcer	10
e00032	Jedi Knight	40

# Janitors Table

---

This table will contain a list of all janitors and the data pertaining to them. The primary key is EID which stands for Employee ID.

---

## Functional Dependencies

EID → ToolOfChoice

---

## SQL Create Statement

```
create table janitors (  
    EID char(10) not null,  
    ToolOfChoice varchar(30) not null,  
    primary key(EID),  
    foreign key(EID) references employees(EID)  
)
```

---

## Sample Data

EID	ToolOfChoice
e00013	Broom
e00027	Mop
e00031	Rake
e00032	Light Saber
e00033	Broom
e00035	Trash Can
e00037	Shovel

# Caretakers Table

---

This table will contain a list of caretakers and the data that pertains to them. The primary key is EID which stands for Employee ID.

---

## Functional Dependencies

EID → FavoriteAnimal

---

## SQL Create Statement

```
create table caretakers (  
    EID char(10) not null,  
    FavoriteAnimal varchar(30),  
    primary key(EID),  
    foreign key(EID) references employees(EID)  
)
```

---

## Sample Data

EID	FavoriteAnimal
e00001	Penguin
e00008	Dolphin
e00010	Monkey
e00015	Horse
e00018	Rabbit
e00019	Fox
e00029	Gorilla
e00030	Parrot
e00034	Snake

## Directors Table

---

This table will contain a list of directors and the data that describes them. The primary key is EID which stands for Employee ID.

---

### Functional Dependencies

EID → NumOfShowsDirected, MaxNumOfShowsPerWeek

---

### SQL Create Statement

```
create table directors (  
    EID char(10) not null,  
    NumOfShowsDirected int not null,  
    MaxNumOfShowsPerWeek int not null,  
    primary key(EID),  
    foreign key(EID) references employees(EID)  
)
```

---

### Sample Data

EID	NumOfShowsDirected	MaxNumOfShowsPerWeek
e00003	70	4
e00004	15	1
e00011	40	2
e00012	100	1
e00017	50	3

# Tourguides Table

---

This table will contain a list of tour guides and the data pertaining to them. The primary key is EID which stands for Employee ID.

---

## Functional Dependencies

EID → MaxHrsPerWeek, NumToursLed, YearsWithCompany

---

## SQL Create Statement

```
create table tourguides (  
    EID char(10) not null,  
    MaxHrsPerWeek int not null,  
    NumToursLed int not null,  
    YearsWithCompany int,  
    primary key(EID),  
    foreign key(EID) references employees(EID)  
)
```

---

## Sample Data

EID	MaxHrsPerWeek	NumOfToursLed	YearsWithCompany
e00006	10	15	2
e00012	40	200	20
e00016	30	75	8
e00021	15	300	5
e00023	25	125	4
e00036	25	75	2
e00040	20	90	4

# Species Table

---

This table will contain a list of valid species that the animals can belong to. The primary key is SpeciesCode which is a list of the kinds of species of animals.

---

## Functional Dependencies

SpeciesCode →

---

## SQL Create Statement

```
create table species (  
    SpeciesCode char(15) not null,  
    primary key(SpeciesCode)  
)
```

---

## Sample Data

SpeciesCode
Amphibian
Arthropod
Bird
Fish
Mammal
Reptile

# Environments Table

---

This table will contain a list of types of environments and the data that describes them. The primary key is EnvID which stands for Environment ID.

---

## Functional Dependencies

EnvID → EnvironmentType, WorldRegion, AvgTempDegF

---

## SQL Create Statement

```
create table environments (  
    EnvID char(10) not null,  
    EnvironmentType varchar(30) not null,  
    WorldRegion varchar(30) not null,  
    AvgTempDegF int not null,  
    primary key(EnvID)  
)
```

---

## Sample Data

EnvID	EnvironmentType	WorldRegion	AvgTempDegF
env001	Arctic	Antarctica	20
env002	Desert	Africa	95
env003	Grassland	North America	75
env004	Ocean	Ocean	65
env005	Tropical Forest	South America	90
env006	Wetland	Europe	80
env007	Mountain	Asia	70

# Animals Table

---

This table will contain a list of all the animals and the data pertaining to them. The primary key is AID which stands for Animal ID.

---

## Functional Dependencies

AID → Animal, Species, AnimalName, Gender, Age, CareTaker, MaxTempDegF, Environment

---

## SQL Create Statement

```
create table animals (  
    AID char(10) not null,  
    Animal varchar(50) not null,  
    Species char(15) not null,  
    AnimalName varchar(50) not null,  
    Gender char(1) check(Gender='M' or Gender='F'),  
    Age int,  
    CareTaker char(10) not null,  
    MaxTempDegF int not null,  
    Environment char(10) not null,  
    primary key(AID),  
    foreign key(Species) references species(SpeciesCode),  
    foreign key(CareTaker) references caretakers(EID),  
    foreign key(Environment) references environments(EnvID)  
)
```

## Sample Data

AID	Animal	Species	AnimalName	Gender	Age	Caretaker	MaxTempDegF	Environment
a00002	Polar Bear	Mammal	Coca Cola	M	5	e00001	40	env001
a00003	Snowy Owl	Bird	Hedwig	M	3	e00001	65	env001
a00005	Wolverine	Mammal	Logan	M	10	e00018	65	env001
a00006	Penguin	Bird	Linux	M	7	e00001	45	env001
a00012	Golden Eagle	Bird	America	F	24	e00030	100	env002
a00015	Gecko	Reptile	Gecko	M	6	e00034	105	env002
a00016	Roadrunner	Bird	Acme	F	8	e00030	105	env002
a00025	Anaconda	Reptile	Snoop Dogg	M	5	e00034	100	env003
a00026	Horse	Mammal	Mr. Ed	M	12	e00015	85	env003
a00032	Shark	Fish	Bruce	M	32	e00008	75	env004
a00038	Angler	Fish	Bright Eyes	F	5	e00001	70	env004
a00039	Turtle	Amphibian	Leonardo	M	42	e00029	80	env004
a00040	Gorilla	Mammal	Buddy	M	22	e00010	100	env005
a00041	Giant Panda	Mammal	Bamboo	F	24	e00010	100	env005
a00042	Chimpanzee	Mammal	Marcel	M	2	e00010	95	env005
a00045	Lynx	Mammal	Sausage	M	9	e00019	100	env005
a00051	Frog	Amphibian	Kermit	M	19	e00029	85	env006
a00057	Hedgehog	Mammal	Sonic	M	7	e00019	80	env007
a00058	Lion	Mammal	Simba	M	14	e00015	90	env007
a00059	Horned Owl	Bird	Who	F	21	e00030	85	env007
a00060	Red Fox	Mammal	Marist	M	11	e00019	85	env007

# SecurityDuties Table

---

This table will contain a list of all security guards and the environments that they patrol. The primary keys are SecurityGuard which will stand for the Employee ID of a security guard and Environment which will stand for the Environment ID.

---

## Functional Dependencies

SecurityGuard, Environment →

---

## SQL Create Statement

```
create table securityduties (  
    SecurityGuard char(10) not null,  
    Environment char(10) not null,  
    primary key(SecurityGuard, Environment),  
    foreign key(SecurityGuard) references security(EID),  
    foreign key(Environment) references environments(EnvID)  
)
```

---

## Sample Data

SecurityGuard	Environment
e00007	env006
e00022	env003
e00022	env007
e00025	env001
e00026	env005
e00026	env006
e00028	env002
e00028	env007
e00032	env004

## Shows Table

---

This table will contain a list of the entertainment shows and the data that pertains to them. The primary key is ShowID which stands for the Show ID.

---

### Functional Dependencies

ShowID → NameOfShow, Director, Environment, TimeStartArmy, DescriptionOfShow

---

### SQL Create Statement

```
create table shows (  
    ShowID          char(10)    not null,  
    NameOfShow      varchar(50) not null,  
    Director        char(10)    not null,  
    Environment     char(10)    not null,  
    TimeStartArmy   int          not null,  
    DescriptionOfShow varchar(250),  
    primary key>ShowID),  
    foreign key(Director) references directors(EID),  
    foreign key(Environment) references environments(EnvID)  
)
```

---

### Sample Data

ShowID	NameOfShow	Director	Environment	TimeStartArmy	Description
s00001	Ice Capades	e00003	env001	14	Artic animals on ice
s00002	Lion King	e00011	env003	16	Lion King brought to you in real life
s00003	Cats	e00012	env003	10	These are no ordinary house cats
s00004	Anaconda	e00012	env005	12	Featuring our slithering friends
s00005	Skies R Us	e00017	env007	14	The land is ruled by people, but the sky is ruled by us!
s00006	Oceanography	e00004	env004	11	Explore the undersea world

# ShowCast Table

---

This table will contain a list of the animals that will appear in each show. The primary keys are ShowID which stands for the Show ID and Animal which will stand for each Animal ID.

---

## Functional Dependencies

ShowID, Animal →

---

## SQL Create Statement

```
create table showcast (  
    ShowID          char(10)    not null,  
    Animal          char(10)    not null,  
    primary key(ShowID, Animal),  
    foreign key(ShowID) references shows(ShowID),  
    foreign key(Animal) references animals(AID)  
)
```

---

## Sample Data

ShowID	Animal
s00001	a00001
s00001	a00005
s00002	a00058
s00002	a00061
s00002	a00062
s00004	a00029
s00005	a00003
s00005	a00059
s00006	a00004

## TourDetails Table

---

This table will contain a list of the kinds of tours and the data that describes them. The primary key is TourID which stands for Tour ID.

---

### Functional Dependencies

TourID → LengthOfTourInHrs, MaxNumGuests, TourDescription

---

### SQL Create Statement

```
create table tourdetails (  
    TourID                char(10)    not null,  
    LengthOfTourInHrs    decimal(2,1) not null,  
    MaxNumGuests         int,  
    TourDescription      varchar(250),  
    primary key(TourID)  
)
```

---

### Sample Data

TourID	LengthOfTourInHrs	MaxNumOfGuests	TourDescription
t00001	2.5	50	General walkthrough of the zoo
t00002	1.0	15	Drivethrough the zoo on special SUVs
t00003	4.0	10	Get a detailed view of every environment
t00004	1.0	15	Get a special under the sea view of the Ocean environment
t00005	1.5	15	Take a hike up the Mountain environment
t00006	1.0	15	Frollic in the open Grasslands
t00007	3.0	15	Build a snowfort in the Arctic
t00008	1.0	15	Play around in the mud of the Wetlands
t00009	1.5	15	Hang out in the trees of the Tropical Forest

# Tours Table

---

This table will contain a list of tours and the tour guides that will be leading them. The primary keys are TourID which stands for the Tour ID and TourGuide which stands for the Employee ID of a tour guide.

---

## Functional Dependencies

TourID, TourGuide →

---

## SQL Create Statement

```
create table tours (  
    TourID          char(10)    not null,  
    TourGuide      char(10)    not null,  
    primary key(TourID, TourGuide),  
    foreign key(TourID) references tourdetails(TourID),  
    foreign key(TourGuide) references tourguides(EID)  
)
```

---

## Sample Data

TourID	TourGuide
t00001	e00023
t00002	e00036
t00003	e00012
t00004	e00040
t00005	e00006
t00006	e00006
t00007	e00021
t00008	e00021
t00009	e00040

# JanitorDuties Table

---

This table will contain a list of the janitors and the environments they clean. The primary keys are Janitor which stands for the Employee ID of a janitor and Environment which stands for a Environment ID.

---

## Functional Dependencies

Janitor, Environment →

---

## SQL Create Statement

```
create table janitorduties (  
    Janitor          char(10)    not null,  
    Environment      char(10)    not null,  
    primary key(Janitor, Environment),  
    foreign key(Janitor) references janitors(EID),  
    foreign key(Environment) references Environments(EnvID)  
)
```

---

## Sample Data

Janitor	Environment
e00013	env006
e00027	env003
e00031	env005
e00032	env002
e00033	env006
e00035	env003
e00035	env007
e00037	env004
e00039	env007

# Reports

---

This section shows sample reports based on the data stored in the Poughkeepsie Zoo database. Each report comes with the SQL state to query it, a description of the usage and some sample data of the result.

---

## Student Reimbursement For Cashiers

This report is to be used to reimburse all cashiers for school tuition. Those that apply must be a cashier, they must be attending school, and are currently working more than 20 hours a week.

---

### SQL Query

```
select firstname as 'First Name', lastname as 'Last Name', Address, City, State, Zip
from cashiers c, employees e
where c.eid = e.eid and
attending school = 'Y' and
maxnumhrsperweek > 20
```

---

### Sample Results

First Name	Last Name	Address	City	State	Zip
Amy	Gotti	427 Hill Dr	Bohemia	NY	11681
Denver	Thomas	50 Parker Ave	Poughkeepsie	NY	12601
Julie	Powers	69 Shagadelic Dr	Mountainville	MN	68451

# Tours Per Day

---

This report will show each tour guide and the total number of tours that each of them will lead each day.

---

## SQL Query

```
select e.firstname as 'First Name', e.lastname as 'Last Name', count(t.tourguide) as 'Total Current Tours'  
from employees e, tours t  
where e.eid = t.tourguide  
group by firstname, lastname  
order by 'Total Current Tours' desc
```

---

## Sample Results

First Name	Last Name	Tour Current Total
Britney	Spears	3
Phoebe	Banana-Hammock	2
Robert	Cannistra	2
Muffin	Man	2
Curtis	Martin	1
Michael	Myers	1
Daniel	Clapper	1

# Number of Animals Per Caretaker

---

This report will show each caretaker and how many animals they are in charge of taking care of.

---

## SQL Query

```
select e.firstname as 'First Name', e.lastname as 'Last Name', count(a.caretaker) as 'Number of Animals'  
from employees e, caretakers c, animals a  
where e.eid = c.eid and  
e.eid = a.caretaker  
group by e.firstname, e.lastname  
order by 'Number of Animals' desc
```

---

## Sample Results

First Name	Last Name	Number of Animals
Michelle	Drushy	13
Alan	Labouseur	9
Debbie	Door	9
Ramsey	Young	6
Kerri	Dempsey	6
Steve	Dave	5
Matthew	Maul	5
John	Stamos	5
George	Jungle	4

# Views

---

This section shows sample views that employees of the Poughkeepsie Zoo might or might not have access to. Each section comes with a description of the view, the SQL statement to create the view, and sample queries and results on that view.

---

## ShowDirectory View

This view is to be used by anyone who would like to find out any information on any of the shows and attractions here at the Poughkeepsie Zoo. The view is useful for accessing data about the shows, show times, directors, environments the shows are in, and the number of animals participating.

---

### SQL View Create Statement

create view ShowDirectory

(ShowID, NameOfShow, ShowTimeArmy, FirstName, LastName, EnvironmentType, NumberofAnimals) as

```
select s.showID, s.nameofshow, s.timestartarmy, e.firstname, e.lastname, en.environmenttype, count(sc.animal)
from shows s, showcast sc, employees e, environments en
where e.EID = s.director and
s.environment = en.ENVID and
s.showid = sc.showid
group by s.showid, s.nameofshow, s.timestartarmy, e.firstname, e.lastname, en.environmenttype
```

## ***Shows Directed by Robert Cannistra***

```
select *  
from showdirectory  
where firstname = 'Robert' and  
lastname = 'Cannistra'
```

---

### **Sample Results**

ShowID	ShowName	ShowTimeArmy	FirstName	LastName	Environment	NumberOfAnimals
s00003	Cats	10	Robert	Cannistra	Grassland	4
s00004	Anaconda	12	Robert	Cannistra	Tropical Forest	5

---

## ***Shows by Starting Time***

```
select *  
from showdirectory  
Order by showtimearmy
```

---

### **Sample Results**

ShowID	ShowName	ShowTimeArmy	FirstName	LastName	Environment	NumberOfAnimals
s00003	Cats	10	Robert	Cannistra	Grassland	4
s00006	Oceanography	11	Ben	Affleck	Ocean	15
s00004	Anaconda	12	Robert	Cannistra	Tropical Forest	5
s00005	Skies R Us	14	Andy	Dick	Mountain	8
s00001	Ice Capades	14	Stan	Lee	Arctic	5
s00002	Lion King	16	Marshall	Mathers	Grassland	10

## Caretaker View

This view is to be used by anyone who works at the zoo as a caretaker. The view is useful for accessing data about caretakers and their duties.

---

### SQL View Create Statement

```
create view CareTakerInfo (EID, FirstName, LastName, NumberOfAnimals, FavoriteAnimal) as
  select e.eid, e.firstname, e.lastname, count(a.aid), c.favoriteanimal
  from employees e, animals a, caretakers c
  where e.eid = c.eid and
  a.caretaker = c.eid
  group by e.eid, e.firstname, e.lastname, c.favoriteanimal
```

### *Caretakers by Last Name*

```
select *
from caretakerinfo
order by lastname desc
```

---

### Sample Results

EID	FirstName	LastName	NumberOfAnimals	FavoriteAnimal
e00018	John	Stamos	5	Rabbit
e00029	Matthew	Maul	5	Gorilla
e00008	Alan	Labouseur	9	Dolphin
e00010	George	Jungle	4	Monkey
e00034	Debbie	Door	9	Snake
e00030	Kerri	Dempsey	6	Parrot

# Stored Procedure

---

This section shows a sample stored procedure that the Poughkeepsie Zoo might need to use frequently. The section comes with a description of the stored procedure, the SQL statement, and some sample results.

---

## WeatherResistant Stored Procedure

This stored procedure is used to test whether or not the animals that are participating in certain shows in the environments are able to withstand the temperatures of those environments. By using the data the directors of those shows know which show to take which animal out of because of environmental constraints.

---

### SQL Create Stored Procedure Statement

```
create procedure WeatherResistant as
select s.nameofshow as 'Show Name', a.animalname as 'Animal Name'
from animals a, showcast sc, shows s, environments e
where a.aid = sc.animal and
sc.showid = s.showid and
s.environment = e.enuid and
a.maxtempdegf < e.avgtempdegf
```

### SQL Stored Procedure Execution Statement

```
exec WeatherResistant
```

---

### Sample Results

Show Name	Animal Name
Skies R Us	Hedwig
Skies R Us	Linux
Oceanography	Tusky
Oceanography	Linux

# Security Settings Per Table

---

This section will break down each table and show which groups of users would have access to which tables. Each section will describe a group of users and the tables that they might have access to.

---

## Managers

- Each specific group manager would have access to his or her own table. (i.e. Cashier Manager to the Cashier's table)
- The zoo manager will be the only one allowed to have full access to all tables.

## Visitors

- Visitors might be given access to the animals table to see what kind of animals they might want to visit.
- They might also be given the ability to view the shows table to see what attractions they might want to see.
- They would be given access to the tours table to see the types of tours offered at the zoo.

## Security

- The security force would be given access to the tour, janitor, and caretaker tables so they could reference where people are at a given moment.
- They would also need access to the animals and environments table so they would know what equipment they might need in case of an emergency.
- They also would have access to the security and securityduties tables.

## Caretakers

- Caretakers would need access to the animals table and the environments table to be able to check up on their animals.
- They might also need to be able to see the shows and showcast tables to help prepare their animals for an upcoming performance.
- The caretakers would also be granted access on the caretakers table.
- The caretakers would also have access to the species table so they know what species of animals they are caring for.

## **Directors**

- Directors would need to have access to the directors, shows, and showcast tables.
- They might also need to have the environments and animals table to know where and who they are working with.

## **Cashiers**

- Cashiers would only have access on the cashiers table.

## **Tour Guides**

- TourGuides would have access to the tourguides, tourdetails, and tours table.
- They also would have access to the environments so they would know where they are visiting
- They would need access to the animals and species tables so they can explain what types of animals they would be seeing.
- They also might be given access to the shows table so they could pitch the attractions here at the Poughkeepsie Zoo.

## **Janitors**

- Janitors would need access to the janitors and janitorduties table.
- They also might require access to the environments table to know where they would have to go.

# Problems

---

This section breaks down the known and future problems as well as possible enhancements to the database with either the implementation or the system as a whole.

---

## Known

The database incurred the following problems as of 12/14/2004:

- Lack of security implementation – the security rules have been laid out but have yet to be implemented
  - Lack of creative views – the views on the system are novice at best, future views would increase productivity
  - Lack of simplicity – the database is in BCNF but there are a few tables that contain data that might not be accessible; might need to look into addressing null values or removing unused data
- 

## Future Enhancements

The database will plan to increase these features:

- Adding more views to ensure some security on the table as well as to facilitate the acquisition of data
- Adding more stored procedures to increase productivity from day to day activities
- Adding a trigger to let the user know when they are placing an animal into a particular show that the animal will not be suited well in the environment