

PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage, Bengaluru 560085

Department of Computer Science and Engineering

Jan – May 2020

UE18CS252

Database Management Systems

Project Report

Zoo Management System

PES1201801770 Srinidhi S

4th Sem Sec. [A] Roll No. 51

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
| **PROJECT SUMMARY**  The Zoo Management System facilitates the Customer to enquire about the wild life and visiting details available on the basis of location and timings, Booking and Cancellation of tickets, enquire about the status of the booked ticket, etc. Here animal loving people search for an animal and get the whole info, they can even search with their availability at the zoo. Transactions include action functions for triggers, all insert, update and delete processes. Transaction maintains all ACID properties. Zoos were initially started for the entertainment of people. Gradually over the years, they have come to play an important role in conservation.  So, as a part of conservation, project includes a trigger for any update or insert of an animal kind with population less than 5, it is going to alert with an exception. Queries includes retrieval of customer, employee details, Aggregate functions on attributes like salary etc.  For security purposes, the access control measures have been taken, Database uses GRANT statements to privilege access. Holds valid for both Non-additive(lossless) join and Dependency Preserving properties.  CAPABILITIES:  Easy to use and handle.  Booking tickets from anywhere as per the provided details and timings.  All guidance option should be available beforehand.  User should get detailed information about every animal, employees and their booking.  System should be able to update at regularly intervals. |

[**Introduction**](#_gwqiqapyxr0x) **2**

[**Data Model**](#_p451stoz4u81) **2**

[**FD and Normalization**](#_wc0v2611q54c) **2**

[**DDL**](#_pn441yprtk2f) **3**

[**Triggers**](#_a9bswny9ffj) **3**

[**SQL Queries**](#_exvna2br67rz) **3**

[**Conclusion**](#_k6bbxckcf51j) **3**

# Introduction

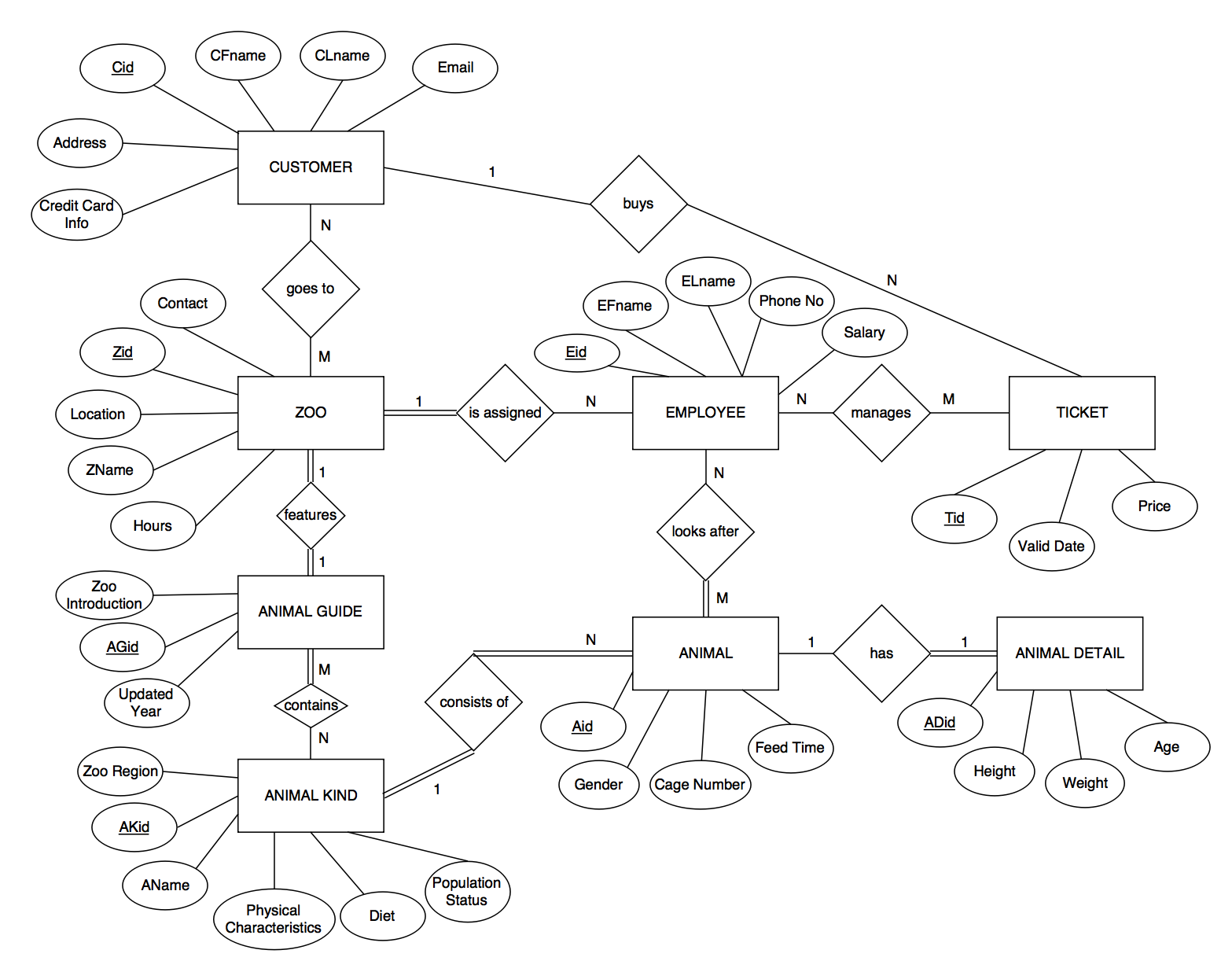
The Zoo Management System facilitates the Customer to enquire about the wild life and visiting details available on the basis of location and timings, Booking and Cancellation of tickets, enquire about the status of the booked ticket, etc.

* Various entities identified in the system are Customer, Employee, Ticket, Zoo, Animal Guide, Animal kind, Animal detail etc. The entity and its attributes are describing the relation between each other.

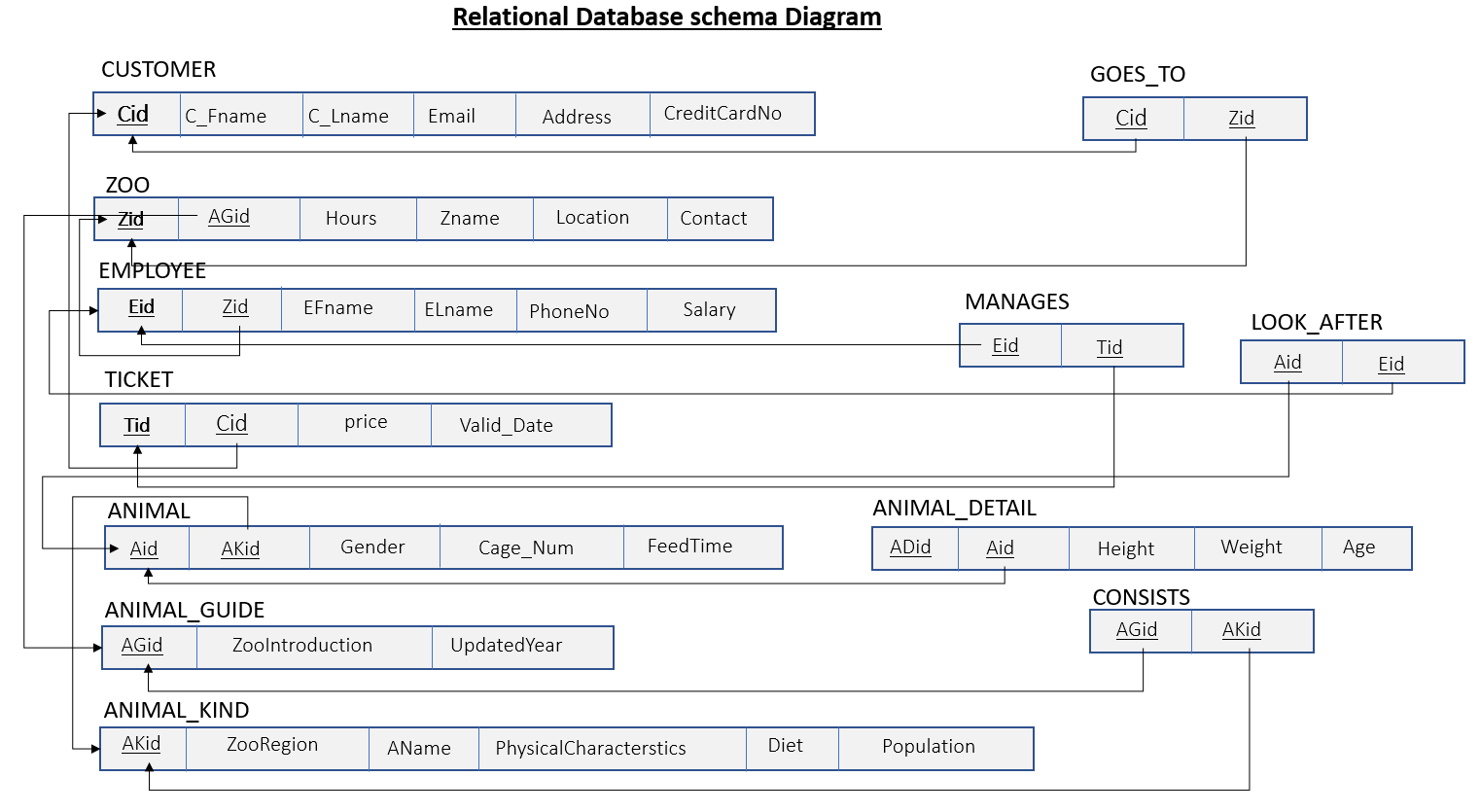
|  |  |  |
| --- | --- | --- |
| **Object** | **Relation** | **Object** |
| **CUSTOMER** | Books  Cancels | Tickets  Tickets |
| **ZOO** | Has  Has | Animal  Employee |
| **EMPLOYEE** | Handle  Serves(tickets) | Animal  Customer |
| **ANIMALS** | Have | Employee |
|  |  |  |

# Data Model

**Entity Relation Diagram**



**Relation Schema Diagram**



**Keys :**

PRIMARY KEYS

Cid = Customer Id, being unique to a customer.

Zid = Zoo Id, unique to the zoo.

Eid = Employee Id unique to a employee.

Tid = Ticket Id unique to a ticket.

Aid = Animal Id for a specific animal.

AGid = Animal Guide specific for a Zoo

AKid = Animal Kind id for a specific species.

ADid = Animal detail id for a specific animal of a kind.

FOREIGN KEYS

Cid, Zid, Aid, AKid, Tid. And the same set of keys make the **Candidate Keys**

# FD and Normalization

Functional Dependencies

Cid {C\_Fname , C\_Lname , Email , Address , CreditcardNo}

{Zid , AGid} {Hours , Zname , Location , Contact}

{Eid , Zid} {EFname , ELname , PhoneNo , Salary }

{Tid , Cid} { Price , ValidDate}

{Aid , AKid} {Gender , CageNumber , FeedTime}

{AGid} { ZooIntroduction , UpdatedDate}

{AKid} { ZooRegion , AName , PhysicalCharacteristics , Diet , population }

{ADid , Aid} {Height , Weight , Age}

Normal Forms

1. BCNF:

The relations GOES\_TO, MANAGES, CONSISTS\_OF is consisting of 2 candidate keys each thus represents the BCNF (Boyce Codd Normal Form).

1. 1NF :

Since in all relations no multivalued attributes are seen or in other way attributes in every relation are atomic.

1. 2NF :

There is no partial dependency in functional dependencies, thus exhibits 2NF. Ex: {Eid , Zid} {EFname , ELname , PhoneNo , Salary }, here RHS of FD depends on both Employee Id and Zoo Id. Etc. Some violations are discussed at the end.

1. 3NF :

Here No non-prime attributes determines the other non-prime attribute, so we don’t have transitive dependency

VIOLATION OF 2NF:

Violation of 3NF: In ANIMAL\_DETAIL relation ADid Aid,

Aid { Gender , CageNumber , FeedTime } and thus ADid determines { Gender , CageNumber , FeedTime } Thus gives the transitive dependency .

Violation of 2NF:

1. {ADid , Aid} {Height , Weight , Age}

{ADid } {Height , Weight , Age} is valid .

2. {Zid , AGid} {Hours, Zname, Location, Contact}

Zid {Hours, Zname, Location, Contact} is also valid

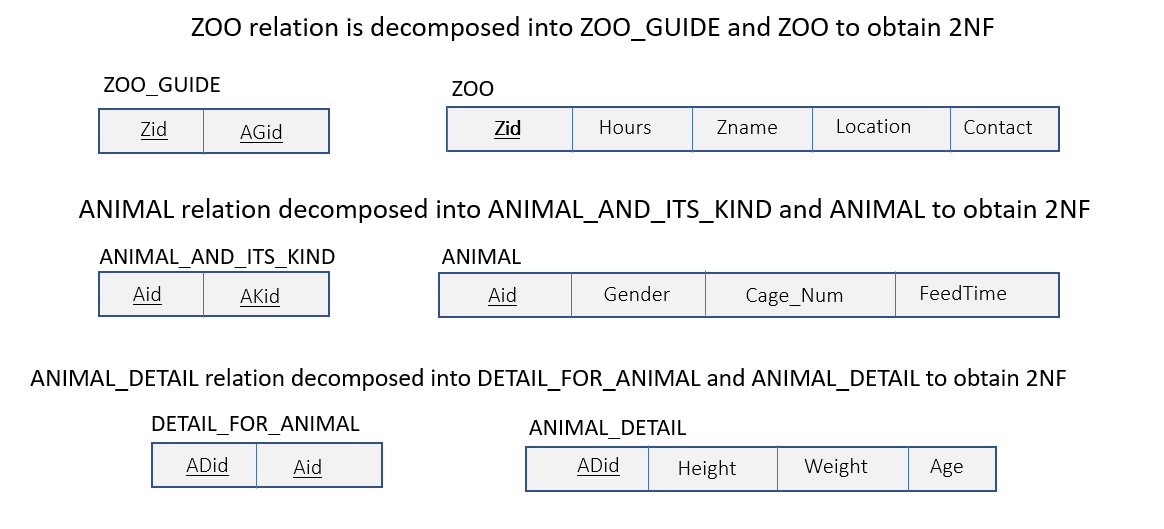
3. {Aid , AKid} {Gender , CageNumber , FeedTime}

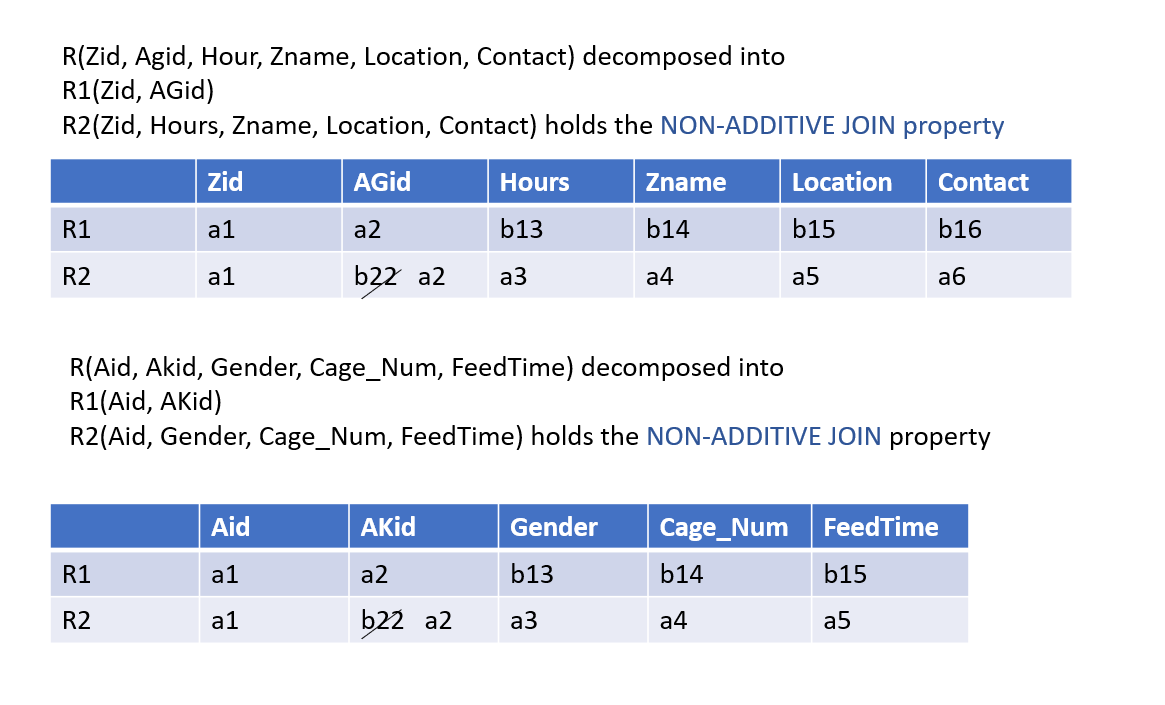
{Aid} { Gender , CageNumber , FeedTime }

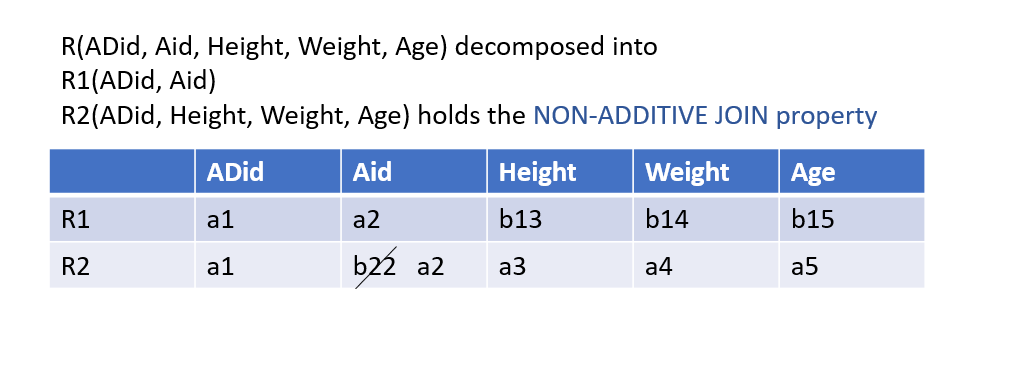
Aid alone can determine the RHS of FD, is also valid Functional dependency

Thus, shows Partial dependency.

Applying Normalization remedies for ZOO, ANIMAL, ANIMAL\_DETAIL relations to obtain 2NF, we get following decomposed relations D.



**FOR NON-ADDITIVE(LOSSLESS) JOIN PROPERTY:**



And these decompositions are **Dependency preserving** also. Because all FDs are obtainable.

# DDL

DROP DATABASE IF EXISTS WILDLIFE;

CREATE DATABASE WILDLIFE;

\c wildlife

DROP TABLE IF EXISTS CUSTOMER CASCADE;

DROP TABLE IF EXISTS ZOO CASCADE;

DROP TABLE IF EXISTS EMPLOYEE CASCADE;

DROP TABLE IF EXISTS TICKET CASCADE;

DROP TABLE IF EXISTS ANIMAL CASCADE;

DROP TABLE IF EXISTS ANIMAL\_GUIDE CASCADE;

DROP TABLE IF EXISTS ANIMAL\_KIND CASCADE;

DROP TABLE IF EXISTS ANIMAL\_DETAIL CASCADE;

DROP TABLE IF EXISTS ZOO\_GUIDE CASCADE;

DROP TABLE IF EXISTS ANIMALS\_AND\_ITS\_KIND CASCADE;

DROP TABLE IF EXISTS DETAIL\_FOR\_ANIMAL CASCADE;

DROP TABLE IF EXISTS LOOK\_AFTER CASCADE;

DROP TABLE IF EXISTS GOES\_TO CASCADE;

DROP TABLE IF EXISTS MANAGES CASCADE;

DROP TABLE IF EXISTS CONSISTS CASCADE;

CREATE TABLE CUSTOMER(

Cid CHAR(5) NOT NULL,

C\_FName VARCHAR(20) NOT NULL,

C\_LName VARCHAR(20) NOT NULL,

Email VARCHAR(50),

Address VARCHAR(100),

CreditCardNo NUMERIC(10,0),

PRIMARY KEY(Cid)

);

CREATE TABLE ZOO (

Zid CHAR(4) NOT NULL,

Hours DECIMAL(2,0) NOT NULL,

Zname VARCHAR(100) NOT NULL,

Location VARCHAR(100) NOT NULL,

Contact NUMERIC(10) NOT NULL,

PRIMARY KEY(Zid)

);

CREATE TABLE EMPLOYEE(

Eid CHAR(10) NOT NULL,

Zooid CHAR(4) NOT NULL,

EFname VARCHAR(30) NOT NULL,

ELname VARCHAR(30) NOT NULL,

PhoneNo NUMERIC(10,0) NOT NULL,

Salary DECIMAL(10,2) NOT NULL CHECK(Salary > 10000),

PRIMARY KEY(Eid),

UNIQUE(Zid),

FOREIGN KEY(Zid) REFERENCES ZOO(Zid)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE TICKET(

Tid INT NOT NULL,

CustId CHAR(5) NOT NULL,

Price INT NOT NULL CHECK(Price>25 ),

Valid\_Date DATE NOT NULL,

PRIMARY KEY(Tid),

UNIQUE(Cid),

FOREIGN KEY(Cid) REFERENCES CUSTOMER(Cid)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE ANIMAL(

Aid CHAR(4) NOT NULL,

Gender CHAR NOT NULL,

Cage\_Num INT NOT NULL,

FeedTime TIME NOT NULL, DEFAULT ‘10:30:00’,

PRIMARY KEY(Aid)

);

CREATE TABLE ANIMAL\_GUIDE(

AGid VARCHAR(13) NOT NULL,

ZooIntroduction VARCHAR(20) NOT NULL,

UpdatedYear CHAR(4) NOT NULL DEFAULT ‘2008’,

PRIMARY KEY(AGid)

);

CREATE TABLE ANIMAL\_KIND(

AKid CHAR(4) NOT NULL,

ZooRegion VARCHAR(20) NOT NULL,

AName VARCHAR(13) NOT NULL,

PhysicalCharacteristics VARCHAR(500) DEFAULT ‘NO INFORMATION AVAILABLE’,

Diet VARCHAR(20),

Population INT NOT NULL,

PRIMARY KEY (AKid)

);

CREATE TABLE ANIMAL\_DETAIL(

ADid CHAR(4) NOT NULL,

Height INT,

Weight INT,

Age INT,

PRIMARY KEY(ADid)

);

CREATE TABLE DETAIL\_FOR\_ANIMAL(

ADid CHAR(4) NOT NULL,

Aid CHAR(4) NOT NULL,

FOREIGN KEY(Aid) REFERENCES ANIMAL(Aid)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE ZOO\_GUIDE(

Zid CHAR(4) NOT NULL,

AGid VARCHAR(30) NOT NULL,

FOREIGN KEY(AGid) REFERENCES ANIMAL\_GUIDE(AGid)

ON DELETE SET NULL ON UPDATE CASCADE,

FOREIGN KEY(Zid) REFERENCES ZOO(Zid)

ON DELETE SET NULL ON UPDATE CASCADE

);

CREATE TABLE ANIMALS\_AND\_ITS\_KIND(

Aid CHAR(4) NOT NULL,

AKid INT NOT NULL,

PRIMARY KEY(Aid),

FOREIGN KEY(AKid) REFERENCES ANIMAL\_KIND(AKid)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE LOOK\_AFTER (

Aid CHAR(4) NOT NULL,

Eid CHAR(10) NOT NULL,

FOREIGN KEY(Aid) REFERENCES ANIMAL(Aid)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(Eid) REFERENCES EMPLOYEE(Eid)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE GOES\_TO(

Cid CHAR(5) NOT NULL,

Zid CHAR(4) NOT NULL,

FOREIGN KEY(Cid) REFERENCES CUSTOMER(Cid)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(Zid) REFERENCES ZOO(Zid)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE MANAGES(

EmpId CHAR(10) NOT NULL,

TktId INT NOT NULL,

FOREIGN KEY(EmpId) REFERENCES EMPLOYEE(Eid)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(TktId) REFERENCES TICKET(Tid)

ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE CONSISTS(

AGid VARCHAR(13) NOT NULL,

AKid INT NOT NULL,

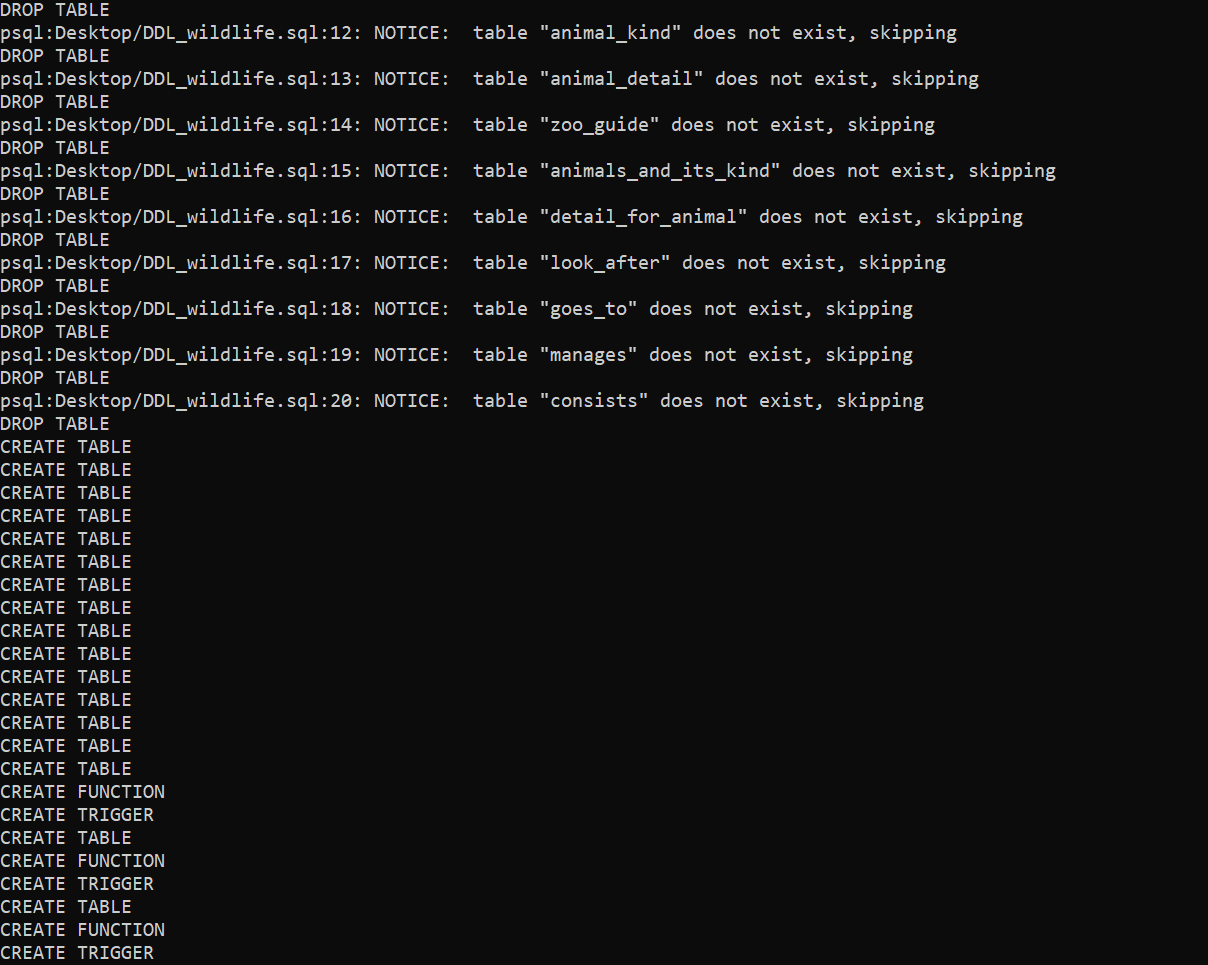
FOREIGN KEY(AGid) REFERENCES ANIMAL\_GUIDE(AGid)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(AKid) REFERENCES ANIMAL\_KIND(AKid)

ON DELETE CASCADE ON UPDATE CASCADE

);



**FOR SECURITY MEASURES :**

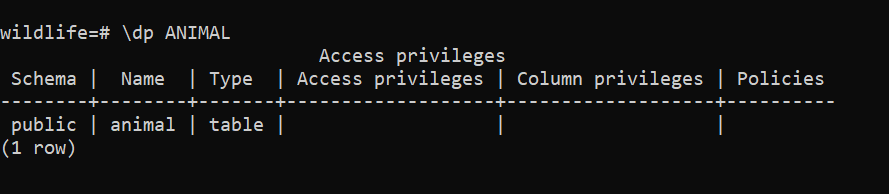
GRANT SELECT ON ANIMAL TO PUBLIC;

GRANT SELECT ON ANIMAL\_KIND TO PUBLIC;

GRANT SELECT ON ZOO TO PUBLIC;

GRANT SELECT ON ANIMAL\_GUIDE TO PUBLIC;

GRANT SELECT, UPDATE, INSERT ON EMPLOYEE TO admin**;**

// Here admin is postgres

# Triggers

1. Since it’s related to wildlife, I thought of wildlife conservation, and when the population of any animal kind falls below 5 in any zoo (on updating the value), the database should give a message of species being ENDANGERED.

SQL STATEMENTS FOR THAT:

CREATE OR REPLACE FUNCTION log\_alert() RETURNS TRIGGER AS $$

BEGIN

IF NEW.Population < 5 THEN

RAISE EXCEPTION 'ALERT ANIMALS BEING ENDANGERED';

END IF;

RETURN current\_timestamp;

END;$$

LANGUAGE PLPGSQL;

CREATE TRIGGER alert

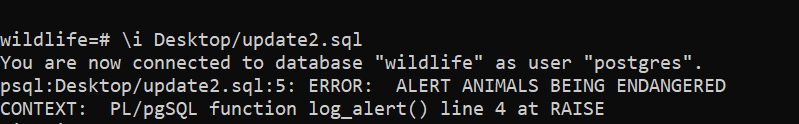
AFTER UPDATE OF POPULATION ON ANIMAL\_KIND

FOR EACH ROW

WHEN (OLD.POPULATION IS DISTINCT FROM NEW.POPULATION)

EXECUTE FUNCTION log\_alert();

OUTPUT: It says alert, and specifies animal being endangered in line 3.



1. A trigger , basically a audit trial on EMPLOYEE on update of salary

CREATE TABLE EMPLOYEE\_AUDIT(

Eid CHAR(10) NOT NULL,

Salary DECIMAL(10,2) NOT NULL,

changed\_on TIMESTAMP(6) NOT NULL

);

CREATE OR REPLACE FUNCTION log\_employee\_update()

RETURNS trigger AS $$

BEGIN

IF NEW.Salary <> OLD.Salary THEN

INSERT INTO EMPLOYEE\_AUDIT(Eid,Salary,changed\_on)

VALUES(OLD.Eid,OLD.Salary,now());

END IF;

RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER check\_update\_on\_employee

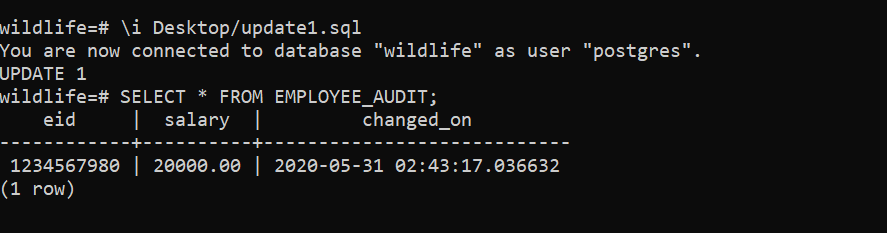
AFTER UPDATE ON EMPLOYEE

FOR EACH ROW

WHEN (OLD.Salary IS DISTINCT FROM NEW.Salary)

EXECUTE FUNCTION log\_employee\_update();

OUTPUT: Gives the old(changed) value of salary and the timestamp.



3. CREATE TABLE EMPLOYEE\_AUDIT\_NAME(

Eid CHAR(10) NOT NULL,

ELname VARCHAR(30) NOT NULL,

changed\_on TIMESTAMP(6) NOT NULL

);

CREATE OR REPLACE FUNCTION log\_employeeDetail\_update()

RETURNS trigger AS $$

BEGIN

IF NEW.ELname <> OLD.ELname THEN

INSERT INTO EMPLOYEE\_AUDIT\_NAME(Eid,ELname,changed\_on)

VALUES(OLD.Eid,OLD.ELname,now());

END IF;

RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER check\_update\_on\_employeeDetail

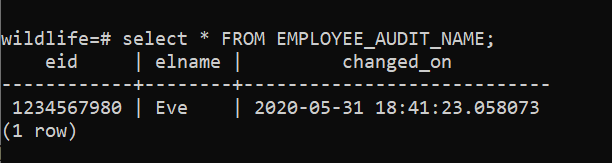
AFTER UPDATE ON EMPLOYEE

FOR EACH ROW

WHEN (OLD.ELname IS DISTINCT FROM NEW.ELname)

EXECUTE FUNCTION log\_employeeDetail\_update();

OUTPUT: Change of last name is audited.



# SQL Queries

1. Retrieve the details of a customer using Customer Id.

SELECT Cid, C\_Fname, C\_Lname, Email, Address

FROM CUSTOMER

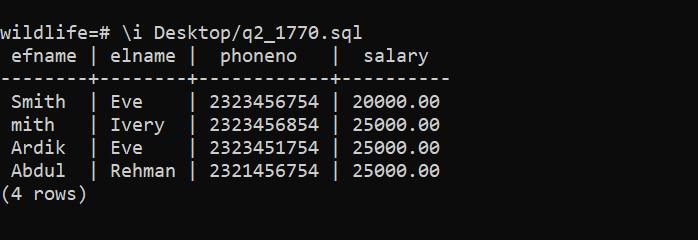


1. Retrieve the employee details from zoo id MYS1, BAN1 using JOIN

SELECT EFname, ELname, PhoneNo, Salary

FROM EMPLOYEE LEFT OUTER JOIN ZOO ON Zid = Zooid

WHERE (Zooid = 'BAN1') OR (Zooid='MYS1');



1. Retrieve the name of employee name and id whose last name is ‘Eve’.

SELECT EMPLOYEE.Eid, EFname, ELname

FROM EMPLOYEE

WHERE EMPLOYEE.Eid IN

( SELECT EMPLOYEE.Eid

FROM LOOK\_AFTER, EMPLOYEE

WHERE LOOK\_AFTER.Empid = EMPLOYEE.Eid AND EMPLOYEE.ELname = 'Eve'

)

OR

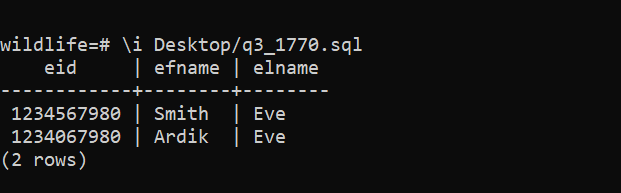
EMPLOYEE.Eid IN

( SELECT EMPLOYEE.Eid

FROM MANAGES, EMPLOYEE

WHERE EmpID = Eid AND EMPLOYEE.ELname = 'Eve'

);

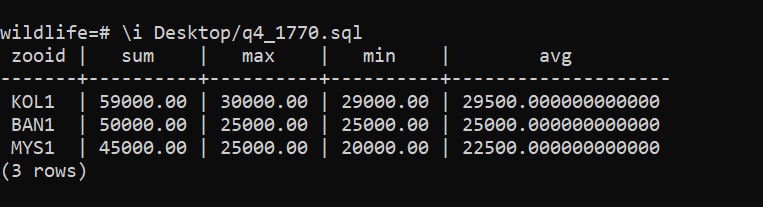


1. Retrieve the sum, average, minimum, maximum amount of salary of an employee and group it by each zoo

SELECT Zooid, SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)

FROM EMPLOYEE

GROUP BY Zooid;

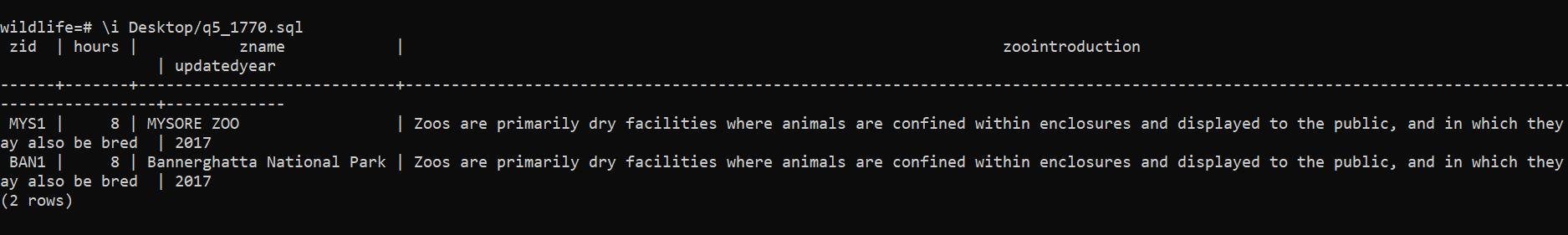


1. Retrieve the Zoo details of the Zoos whose Guide updated on 2017

SELECT ZOO.Zid, ZOO.Hours, ZOO.Zname, ZooIntroduction, UpdatedYear

FROM ZOO, ZOO\_GUIDE, ANIMAL\_GUIDE

WHERE ZOO.Zid = ZOO\_GUIDE.Zid AND ANIMAL\_GUIDE.AGid = ZOO\_GUIDE.AGid AND ANIMAL\_GUIDE.UpdatedYear = '2017'



1. Retrieve the total number of employees working in respective zoo and only if total employee being greater than 2.

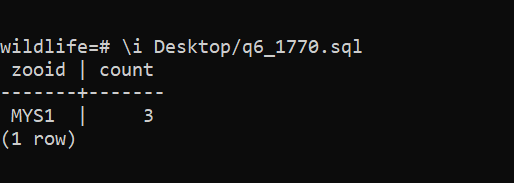
SELECT Zooid, COUNT(\*)

FROM EMPLOYEE, ZOO

WHERE Zooid = Zid

GROUP BY Zooid

HAVING COUNT(\*) > 2;



1. Retrieve the names of employees who look after animal ‘Giraffe’.

SELECT EFname,ELname,Zooid

FROM EMPLOYEE

WHERE EXISTS(

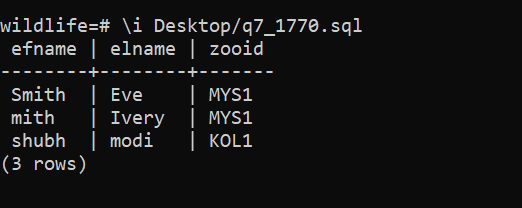
SELECT \*

FROM ANIMAL\_KIND, ANIMALS\_AND\_ITS\_KIND, LOOK\_AFTER

WHERE ANIMALS\_AND\_ITS\_KIND.Aid = LOOK\_AFTER.Aid AND

LOOK\_AFTER.Eid = EMPLOYEE.Eid AND ANIMAL\_KIND.AName = 'Giraffe'

);

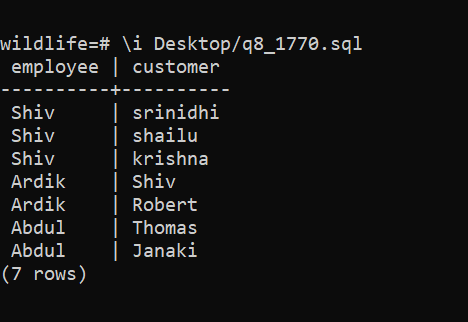


1. Retrieve the name the employee who manages tickets at zoo.

SELECT EMPLOYEE.EFname AS Employee, CUSTOMER.C\_FName AS Customer

FROM MANAGES, TICKET, EMPLOYEE, CUSTOMER

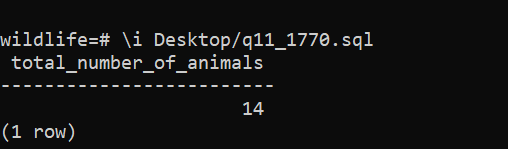
WHERE Tid = TktId AND Eid = EmpId AND Cid = CustId;



1. Retrieve the number of animals in each zoo.

SELECT COUNT(\*) AS TOTAL\_NUMBER\_OF\_ANIMALS

FROM ANIMAL;



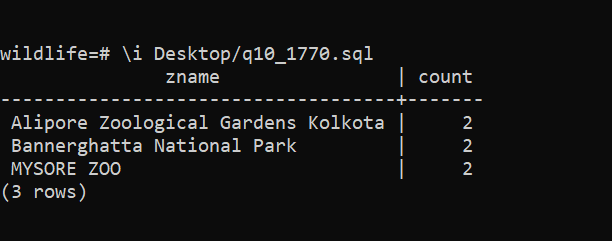
1. Retrieve the number of employees in each zoo.

SELECT Zname,COUNT(\*)

FROM EMPLOYEE,ZOO

WHERE Zid = Zooid

GROUP BY Zid;

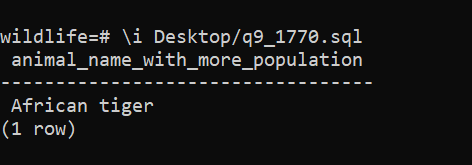


1. Retrieve the animal species whose population is more than 20

SELECT ANIMAL\_KIND.Aname AS Animal\_Name\_with\_More\_population

FROM ANIMAL\_KIND

WHERE Population > 20;



# **Conclusion**

CAPABILITIES:

* Easy to use and handle.
* Booking tickets from anywhere as per the provided details and timings.
* All guidance option should be available beforehand.
* User should get detailed information about every animal, employees and their booking.
* System should be able to update at regularly intervals.

LIMITATIONS:

* System is not very safe for sensitive information like Credit Card Information, money transfer details etc.

FUTURE ENHANCEMENTS:

* It should be secure and reliable.
* A Good Frontend for user-friendly representation.
* Use SQL INJECTION and other cryptography methods for security, user account protection, Access control etc.

**Formatting Guidelines** (make minor changes as required)

font, font size

Something like Calibri, Times New Roman, Verdana, etc.

Size 12, If too small increase to 14; if too big make it 10/11.

Left and right margin say 1 inch.

No. pages – no specific limit but few images could be also be included.