HIGH WALL CORRECTIONAL SOLUTIONS

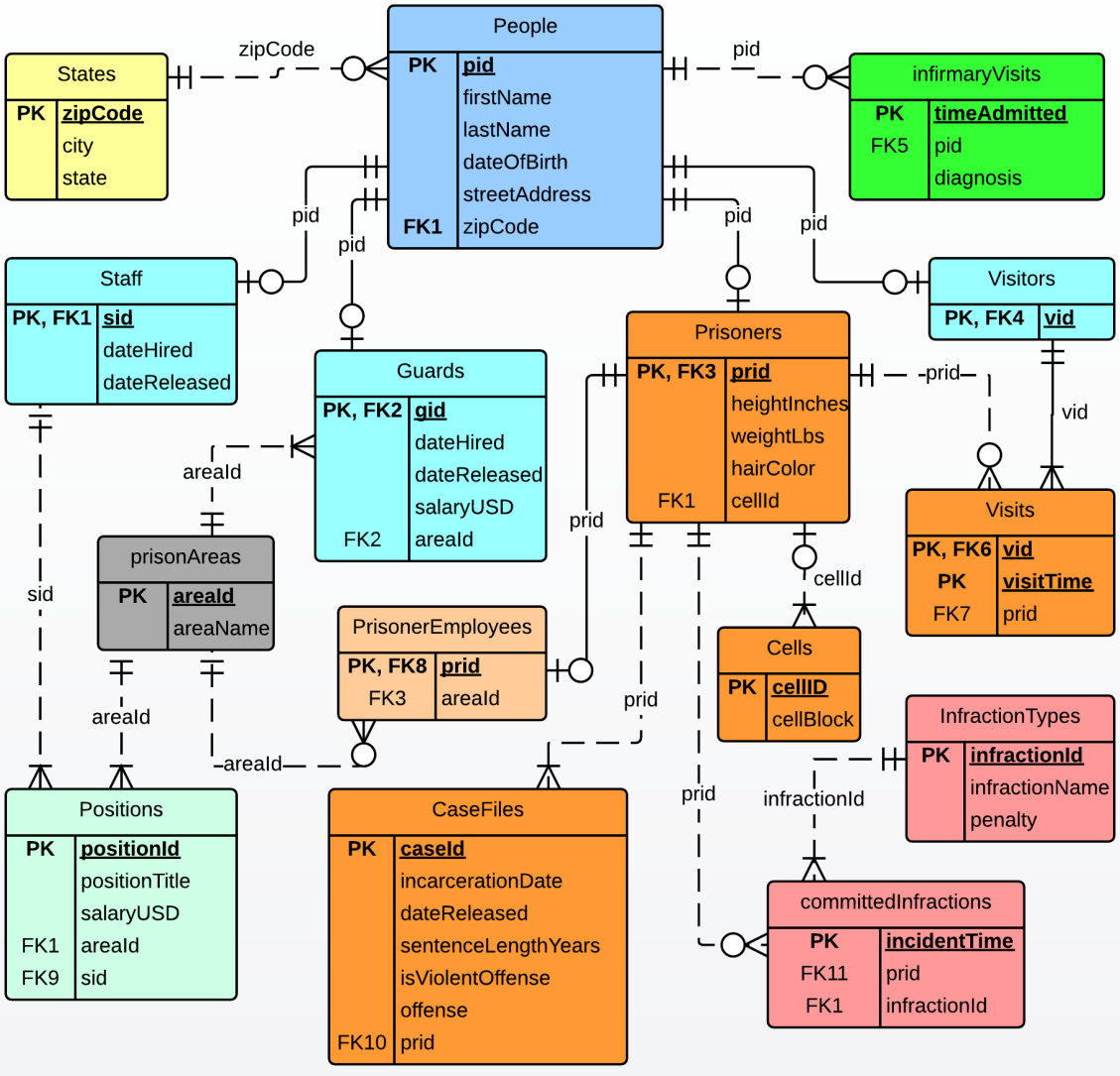
A Database Design Proposal

BY

CODY EICHELBERGER

TABLE OF CONTENTS

This document outlines the structure and entities involved in the design and implementations of a database system for a correctional facility. The purpose of this database is to enable cataloging of the various roles that need to be filled within a prison system, as well as to manage inmate cell assignments, case files, infractions, visitor interactions, employment, other miscellaneous events. This database will allow administration to create useful information from queries that provide valuable statistics and other facts from the catalogued data. By managing the prison with this database implementation, tracking inmates, guards, visitors, and other personnel will be streamlined to ensure accurate and swift record keeping. The ultimate goal is to provide a fully functional, normalized database that will beautifully serve the needs of a correctional institute.


People
PK pid
firstName
lastName
dateOfBirth
streetAddress
FK1 zipCode
States
PK zipCode
city
state
zipCode
infirmaryVisits
PK timeAdmitted
FK5 pid
diagnosis
pid
Staff
PK, FK1 sid
dateHired
dateReleased
Guards
PK, FK2 gid
dateHired
dateReleased
salaryUSD
FK2 areaId
Prisoners
PK, FK3 prid
heightInches
weightLbs
hairColor
FK1 cellId
Visitors
PK, FK4 vid
Visits
PK, FK6 vid
PK visitTime
FK7 prid
prisonAreas
PK areaId
areaName
areaId
sid
prid
cellId
PrisonerEmployees
PK, FK8 prid
FK3 areaId
Cells
PK cellID
cellBlock
InfractionTypes
PK infractionId
infractionName
penalty
infractionId
Positions
PK positionId
positionTitle
salaryUSD
FK1 areaId
FK9 sid
CaseFiles
PK caseId
incarcerationDate
dateReleased
sentenceLengthYears
isViolentOffense
offense
FK10 prid
committedInfractions
PK incidentTime
FK11 prid
FK1 infractionId

PEOPLE lists all people and basic attributes functional dependencies

```
CREATE TABLE people (
  pid            char(4) not null unique,
  firstName      text not null,
  lastName       text not null,
  streetAddress  text not null,
  dateOfBirth    date not null,
  zipCode        integer not null,
 primary key(pid),
 foreign key(zipCode) references states(zipCode)
);
```

functional dependencies

```
pid → firstName, lastName,
       streetAddress, dateOfBirth, zipCode
```

SAMPLE DATA ON FOLLOWING PAGE

pid charac	firstname text	lastname text	streetaddress text	dateofbirth date	zipcode integer
p001	Joseph	Stalin	101 Commie Way	1960-01-05	10199
p002	Ted	Bundy	3 Heads Street	1975-02-06	48223
p003	Charles	Manson	21 Cult Lane	1964-09-15	94612
p004	Bernie	Madoff	404 Ponzi Street	1954-04-13	10199
p005	Al	Capone	1 Gangster Circle	1988-10-21	10199
p006	Ted	Kaczynski	90 Unabomb Terrace	1991-07-10	90052
p007	Adam	Lanza	12 Washington Avenue	1989-06-23	60607
p008	Andrea	Kehoe	68 Bath Street	1971-11-10	10199
p009	John	Gacy	200 Newsome road	1994-12-10	60607
p010	timothy	McVeigh	123 Fake Street	1988-08-28	70113
p011	Jim	jones	399 Jones Way	1947-04-13	38101
p012	Scott	Peterson	22 Grey Avenue	1983-05-05	33952
p013	James	Ray	708 Mystery Street	1970-07-09	38101
p014	Jack	Kevorkian	44567 Assisted Street	1990-04-13	90052
p015	Jeffrey	Dahmer	666 God Lane	1992-04-13	10199
p016	Adam	Jones	45 Exeter Street	1980-03-09	10199
p017	Cody	Eichelberg	22227 Hernando Avenue	1993-03-04	33952
p018	Travis	Crabtree	44332 Conway Avenue	1992-05-21	33952
p019	Jordon	Aroyo	914 Alpine Ave	1994-04-24	33952
p020	Alan	Labouseur	94 Postgres Lane	1995-09-16	60607
p021	Bobby	Hill	5 Arlington Street	1992-02-04	77201
p022	Hank	Hill	55 Propane Circle	1975-01-05	77201
p023	Big	Bird	543 Almost Drive	1982-06-11	10199

STAFF lists all staff members and basic attributes

```
CREATE TABLE staff (
  sid            char(4) not null unique references people(pid),
  dateHired      date not null,
  dateReleased   date,
 primary key(sid)
);
```

functional dependencies

sid → dateHired, dateReleased

sid character(4)	datehired date	datereleased date
p016	2013-01-14	
p017	2014-03-20	
p018	2012-06-19	
p019	2014-07-04	
p020	2014-04-21	

GUARDS lists all guards and basic attributes

```
CREATE TABLE guards (
  gid            char(4) not null unique references people(pid),
  dateHired      date not null,
  dateReleased   date,
  salaryUSD      numeric not null,
  areaId         char(3) not null unique references prisonAreas(areaId),
 primary key(gid),
 foreign key(areaId) references prisonAreas(areaId)
);
```

functional dependencies

gid → dateHired, dateReleased, salaryUSD, areaId

gid character(4)	datehired date	datereleased date	salaryusd numeric	areaid character(3)
p021	2013-01-13		30000	a01
p022	2009-03-11		41000	a03
p023	2010-06-19		40000	a05
p024	2014-07-04		25000	a06
p025	2000-04-21		50000	a07

VISITORS lists all visitors

```
CREATE TABLE visitors (
  vid            char(4) not null unique references people(pid),
 primary key(vid)
);
```

vid character(4)
p026
p027
p028
p029
p030

functional dependencies

vid → N/a

PRISONERS lists all prisoners and basic attributes

```
CREATE TABLE prisoners (
  prid           char(4) not null unique references people(pid),
  heightInches   integer not null,
  weightLbs      integer not null,
  hairColor      text,
  cellId         integer not null unique references cells(cellId),
 primary key(prid),
 foreign key(cellId) references cells(cellId)
);
```

functional dependencies

prid → heightInches, weightLbs, hairColor, cellId

SAMPLE DATA ON FOLLOWING PAGE

prid character(4)	heightinches integer	weightlbs integer	haircolor text	cellid integer
p001	67	130	brown	1
p002	65	120	black	2
p003	70	144	brown	4
p004	68	188	blonde	5
p005	67	133	brown	7
p006	66	156	red	8
p007	69	175	blonde	9
p008	63	129	black	10
p009	66	150	black	11
p010	78	175	brown	17
p011	67	166	brown	12
p012	59	162	brown	13
p013	73	143	blonde	14
p014	75	190	black	15
p015	69	201	red	16

PRISONER EMPLOYEES lists all prisoner Employees and the area

```
CREATE TABLE prisonerEmployees (
  prid          char(4) not null unique references prisoners(prid),
  areaId        char(3) references prisonAreas(areaId),
 primary key(prid),
 foreign key(prid) references prisoners(prid),
 foreign key(areaId) references prisonAreas(areaId)
);
```

prid character(4)	areaid character(3)
p003	a04
p004	a04
p005	a05
p008	a08
p010	a13

functional dependencies

prid → areaId

STATES lists all states and cities associated with zipcodes

```
CREATE TABLE states (
  zipCode       integer not null unique,
  city          text not null,
  state         text not null,
 primary key(zipCode)
);
```

functional dependencies

zipCode → city, state

zipcode integer	city text	state text
12601	Poughkeepsie	New York
33952	Port Charlotte	Florida
90052	Los Angeles	California
94612	Oakland	California
60607	Chicago	Illinois
70113	New Orleans	Louisiana
48223	Detroit	Michigan
10199	New York	New York
38101	Memphis	Tennessee
77201	Houston	Texas

PRISON AREAS lists all prison areas

```
CREATE TABLE prisonAreas (
  areaId         char(3) not null unique,
  areaName       text,
 primary key(areaId)
);
```

functional dependencies

areaId → areaName

areaid charact	areaname text
a01	Administration Offices
a02	Cell Blocks
a03	Infirmary
a04	Kitchen
a05	Cafeteria
a06	East Yard
a07	West Yard
a08	Library
a09	Showers
a10	Commissary
a11	East Tower
a12	West Tower
a13	Laundry

CELLS lists all cells and cell blocks

```
CREATE TABLE cells (
  cellId     integer not null unique
             CHECK (0 < cellId AND cellId < 26),
  cellBlock  char(1) not null CHECK (   cellBlock = 'a'
                                     OR cellBlock = 'b'
                                     OR cellBlock = 'c'
                                    ),
 primary key(cellId)
);
```

functional dependencies

cellId → cellBlock

cellid integer	cellblock character(1)
1	a
2	a
4	a
5	a
6	a
7	a
8	b
9	b
10	b
11	b
12	b
13	b
14	c
15	c
16	c
17	c
18	c

POSITIONS lists all staff positions and basic attributes

```
CREATE TABLE positions (
  positionId     char(4) not null unique,
  positionTitle  text,
  salaryUSD      numeric,
  areaId         char(3) not null references prisonAreas(areaId),
  sid            char(4) not null unique references staff(sid),
 primary key(positionId),
 foreign key(areaId) references prisonAreas(areaId),
 foreign key(sid) references staff(sid)
);
```

functional dependencies

positionId → positionTitle, salaryUSD, areaId, sid

positionid character(4)	positiontitle text	salaryusd numeric	areaid character(3)	sid character(4)
r001	Warden	120000	a01	p020
r002	Head Doctor	70000	a03	p019
r003	Head Chef	60000	a04	p018
r004	Commissary Manager	30000	a10	p017
r005	Laundry Manager	40000	a13	p016

CASE FILES lists all prisoner case files and basic attributes

```
CREATE TABLE caseFiles (
  caseId               char(5) not null unique,
  incarcerationDate    date not null,
  dateReleased         date,
  sentenceLengthYears  integer not null CHECK (sentenceLengthYears > 0),
  isViolentOffense     boolean not null,
  offense              text not null,
  prid                 char(4) not null references prisoners(prid),
 primary key(caseId),
 foreign key(prid) references prisoners(prid)
);
```

functional dependencies

caseId → incarcerationDate, dateReleased, sentenceLengthYears, isViolentOffense, offense, prid

SAMPLE DATA ON FOLLOWING PAGE

caseid character(5)	incarcerationdate date	datereleased date	sentencelengthyea integer	isviolent boolean	offense text	prid chara
c001	1999-01-08		20	t	murdered wife	p001
c002	1994-10-12		25	t	murdered neighbor	p002
c003	1985-05-14		45	t	rape	p003
c004	2002-07-02		15	t	armed robbery	p004
c005	2005-02-03		10	f	drug trafficking	p005
c006	2013-08-05		10	f	piracy	p006
c007	2011-11-09		5	f	grand theft	p007
c008	1992-12-11		20	t	murder	p008
c009	2014-03-13		8	f	conspiracy to commit murder	p009
c010	2012-05-15		15	f	drug manufacture	p010
c011	1980-07-16		150	t	murdered village	p011
c012	2003-07-22		12	f	tax evasion	p012
c013	2010-04-28		7	t	manslaughter	p013
c014	2011-03-29		4	f	hacking	p014
c015	2013-01-14		2	t	assault	p015

INFRACTION TYPES lists all infraction types and basic attributes

```
CREATE TABLE infractionTypes (
  infractionId   char(3) not null unique,
  infractionName text not null,
  penalty        text,
 primary key(infractionId)
);
```

functional dependencies

infractionId → infractionName, penalty

infractionid character(3)	infractionname text	penalty text
i01	Attempted escape	One week of solitary confinement
i02	Assault on a guard or staff	Two weeks of solitary confinement
i03	Assault on fellow inmate	Two weeks of solitary confinement
i04	General insubordination	Revoke yard and commissary privileges
i05	Inciting a riot	Two days of solitary and no yard privileges
i06	murder	Transfer to maximum security facility
i07	possession of contraband	One week of no yard privileges

COMMITTED INFRACTIONS lists all infractions committed by prisoners

```
CREATE TABLE committedInfractions (
  incidentTime   timestamp not null unique,
  infractionId   char(3) not null references infractionTypes(infractionId),
  prid           char(4) not null references prisoners(prid),
 primary key(incidentTime),
 foreign key(prid) references prisoners(prid)
);
```

functional dependencies

incidentTime → infractionId, prid

incidenttime timestamp without time zone	infractionid character(3)	prid character(4)
2010-04-18 12:34:00	i01	p001
2011-05-28 14:20:00	i02	p003
2011-11-11 14:50:00	i03	p008
2009-02-20 14:17:00	i04	p011
2013-03-14 14:31:00	i01	p001
2007-09-19 14:23:00	i07	p001
2008-07-01 14:47:00	i05	p001
2009-05-27 14:35:00	i07	p014
2013-01-13 14:03:00	i07	p004
2012-06-18 14:51:00	i04	p001

INFIRMARY VISITS lists all infirmary visits

```
CREATE TABLE infirmaryVisits (
  timeAdmitted   timestamp not null unique,
  pid            char(4) not null unique,
  diagnosis      text not null,
 primary key(timeAdmitted),
 foreign key(pid) references people(pid)
);
```

timeadmitted timestamp without time zone	pid character(4)	diagnosis text
2013-08-08 09:30:00	p001	herpes
2014-09-04 11:30:00	p018	cut
2012-04-12 12:30:00	p002	measles
2011-05-10 21:30:00	p015	bruising
2013-06-09 08:30:00	p024	fever

functional dependencies

timeAdmitted → pid, diagnosis

VISITS lists all visits between inmates and visitors

```
CREATE TABLE visits (
  vid         char(4) not null references visitors(vid),
  visitTime   timestamp not null unique,
  prid        char(4) not null references prisoners(prid),
 primary key(vid, visitTime),
 foreign key(vid) references visitors(vid),
 foreign key(prid) references prisoners(prid)
);
```

vid character(4)	visittime timestamp without time zone	prid character(4)
p026	2013-05-08 09:30:00	p002
p026	2013-06-08 09:32:00	p002
p026	2013-07-08 09:45:00	p002
p026	2013-08-08 10:38:00	p002
p028	2010-04-18 14:30:00	p005
p030	2012-11-28 13:12:00	p004
p027	2014-10-13 11:34:00	p006
p027	2014-10-15 11:45:00	p006
p027	2014-10-16 10:51:00	p006
p029	2009-01-20 15:38:00	p011

functional dependencies

(vid, visitTime) → prid

VIEW PrisonPopulation lists names and cell assignments of all prisoners in population

```
CREATE VIEW PrisonPopulation AS
  SELECT firstName, lastName, prisoners.cellId as cellNumber, cellBlock as cellBlockLetter
  FROM people
  INNER JOIN prisoners
  ON people.pid = prisoners.prid
  INNER JOIN cells
  ON prisoners.cellId = cells.cellId
  ORDER BY lastName;
```

firstname text	lastname text	cellnumber integer	cellblockletter character(1)
Joseph	Stalin	1	a
Ted	Bundy	2	a
Charles	Manson	4	a
Bernie	Madoff	5	a
Al	Capone	7	a
Ted	Kaczynsk	8	b
Adam	Lanza	9	b
Andrea	Kehoe	10	b
John	Gacy	11	b
Jim	jones	12	b
Scott	Peterson	13	b
James	Ray	14	c
Jack	Kevorkia	15	c
Jeffrey	Dahmer	16	c
timothy	McVeigh	17	c

VIEW CurrentStaff lists names, positions, and the dateReleased of all staff

```
CREATE VIEW CurrentStaff AS
  SELECT positionTitle as position, firstName, lastName, dateReleased
  FROM people
  INNER JOIN staff
  ON people.pid = staff.sid
  INNER JOIN positions
  ON staff.sid = positions.sid
  WHERE dateReleased is null
  ORDER BY position DESC;
```

position text	firstname text	lastname text	datereleased date
Warden	Alan	Labouseur	
Laundry Manager	Adam	Jones	
Head Doctor	Jordon	Aroyo	
Head Chef	Travis	Crabtree	
Commissary Manager	Cody	Eichelberger	

VIEW GuardAreas lists names and area assignments of all guards

```
CREATE VIEW guardAreas AS
  SELECT firstName, lastName, areaName as Area
  FROM people
  INNER JOIN guards
  ON people.pid = guards.gid
  INNER JOIN prisonAreas
  ON guards.areaId = prisonAreas.areaId
  ORDER BY lastName;
```

firstname text	lastname text	area text
Big	Bird	Cafeteria
Tickle	Elmo	West Yard
Bobby	Hill	Administration Offices
Hank	Hill	Infirmary
Toucan	Sam	East Yard

REPORTS Interesting Queries – these are queries that demonstrate the analytical potential of databases. These are mild examples, but nonetheless examples of the kinds of information that one can extrapolate from data.

1. Query to return the percentage of the prison population that is nonviolent

```
SELECT TRUNC (
          CAST(
               ( SELECT COUNT(pid) AS nonViolentCount
                 FROM people
                 INNER JOIN prisoners
                 ON people.pid = prisoners.prid
                 INNER JOIN caseFiles
                 ON prisoners.prid = caseFiles.prid
                 WHERE isViolentOffense = false
                ) as decimal(5,2)
              )
                 /
                 ( SELECT COUNT(prid) AS wholePopulation
                   FROM prisoners
                 )
       * 100
            ) as Percent_Nonviolent
```

percent_nonviolent numeric
47

2. Query to return the percentage of the prison population that is under 25

```
SELECT TRUNC (
          CAST(
               ( SELECT COUNT(pid) AS under25Count
                 FROM people
                 INNER JOIN prisoners
                 ON people.pid = prisoners.prid
                 WHERE date_part('year',age(dateOfBirth)) < 25

               ) as decimal(5,2)
              )
                 /
                 ( SELECT COUNT(prid) AS wholePopulation
                   FROM prisoners
                 )
       * 100
            ) as Percent_Under25
```

percent_under25 numeric
29

2. Query to return the percentage of all violent prisoners that have refrained from committing any institutional infractions. This could be used partially to determine good behavior or not when considering parole.

```
SELECT TRUNC (
          CAST(
               ( SELECT COUNT(pid) AS peacefulViolent
                 FROM people
                 INNER JOIN prisoners
                 ON people.pid = prisoners.prid
                 INNER JOIN caseFiles
                 ON prisoners.prid = caseFiles.prid
                 LEFT OUTER JOIN committedInfractions
                 ON casefiles.prid = committedinfractions.prid
                 WHERE committedInfractions.prid IS NULL AND caseFiles.isViolentOffense = true
               ) as decimal(5,2)
             )
                /
                (
                 SELECT COUNT(prisoners.prid) AS violentOffenders
                 FROM prisoners
                 INNER JOIN caseFiles
                 ON prisoners.prid = caseFiles.prid
                 WHERE isViolentOffense = true
                )
       * 100
            ) as Percent_Peaceful_Violent_Prisoners
```

percent_peaceful_violent_prisoners numeric
37

STORED PROCEDURES **these are stored functions that may be called on to automate statements or conduct calculations automatically instead of needing to structure the query each time it is needed.**

1. STORED PROCEDURE add_prisonerEmployee **this automatically makes a newly created prisoner also a prisoner employee if his case file shows that his current offense was nonviolent upon insertion into the casefiles table.**

```
CREATE OR REPLACE FUNCTION add_prisonerEmployee() RETURNS trigger AS
$BODY$
    BEGIN
        IF NEW.isViolentOffense = false THEN
            INSERT INTO prisonerEmployees (prid) VALUES (NEW.prid);
        END IF;
        RETURN NEW;
    END;
 $BODY$
 LANGUAGE plpgsql;
```

SAMPLE DATA FOR THIS PROCEDURE WILL BE PAIRED WITH THE SAMPLE DATA FOR THE TRIGGER THAT ACTIVATES IT IN THE FOLLOWING SECTION

2. STORED PROCEDURE prisonerVistors **this automatically returns a table of the names of visitors that the input prisoner has had**

```
CREATE OR REPLACE FUNCTION prisonerVisitors (IN prisonerId varchar(4))
 RETURNS TABLE("First Name" text, "LastName" text) AS
$BODY$
BEGIN
   RETURN QUERY SELECT DISTINCT people.firstName as first_name, people.lastname as last_name
                       FROM  people
                       INNER JOIN visitors
                       ON people.pid = visitors.vid
                       INNER JOIN visits
                       ON visits.vid = visitors.vid
                       WHERE visits.prid = prisonerId;
END;
$BODY$
LANGUAGE PLPGSQL;
```

```
select prisonervisitors('p002')
```

prisonervisitors record
(Alex,Davis)

3. STORED PROCEDURE releaseDate **this automatically updates the dateReleased column of the new insert into the casefiles table by adding the sentence to the incarceration date of the particular prisoner**

```
CREATE OR REPLACE FUNCTION add_releaseDate() RETURNS TRIGGER AS
$BODY$
  DECLARE
    sentenceLength integer :=  365 * CAST (new.sentenceLengthYears as INTEGER);
    calcDateReleased TIMESTAMP := new.incarcerationDate + sentenceLength;
    BEGIN


         IF NEW.dateReleased IS NULL THEN
            UPDATE caseFiles
            SET dateReleased = calcDateReleased
            WHERE new.prid = prid ;
        END IF;
        RETURN NEW;
    END;
 $BODY$
 LANGUAGE plpgsql;
```

SAMPLE DATA FOR THIS PROCEDURE WILL BE PAIRED WITH THE SAMPLE DATA FOR THE TRIGGER THAT ACTIVATES IT IN THE FOLLOWING SECTION

4. STORED PROCEDURE prisonerAge **this simply calculates the age of a given prisoner**

```
 CREATE OR REPLACE FUNCTION prisonerAge ( prisonerId varchar(4))
 RETURNS INTERVAL AS
$BODY$
 DECLARE
   birthday date := (SELECT people.dateOfBirth
                     FROM people
                     INNER JOIN prisoners
                     ON people.pid = prisoners.prid
                     WHERE prisoners.prid = prisonerId
                    );
BEGIN
   RETURN age(birthday);
END;
$BODY$
LANGUAGE PLPGSQL;
```

```
select prisonerage('p001')
```

prisonerage interval
54 years 10 mons 30 days

TRIGGERS these call functions upon a specified activity on a certain table such as insert, update, or delete

1. TRIGGER add_prisonerEmployee **this automatically makes a newly created prisoner also a prisoner employee if the isViolentOffense attribute of the inserted file is false.**

```
CREATE TRIGGER add_prisonerEmployee
AFTER INSERT ON caseFiles
FOR EACH ROW
EXECUTE PROCEDURE add_prisonerEmployee();
```

```
INSERT INTO caseFiles ( caseId, incarcerationDate,
                        dateReleased, sentenceLengthYears,
                        isViolentOffense, offense, prid
                      )
 VALUES ( 'c03', '1/1/2014', null, 10, false, 'TESTCRIME', 'p032' );
```

BEFORE INSERT

prid character(4)	areaid character(3)
p003	a04
p004	a04
p005	a05
p008	a08
p010	a13

AFTER INSERT

prid character(4)	areaid character(3)
p003	a04
p004	a04
p005	a05
p008	a08
p010	a13
p032	

2. TRIGGER add_releaseDate **on an insert to caseFiles, this calls the stored procedure add_releaseDate() which updates the dateReleased column of the new insert into the casefiles table by adding the sentence to the incarceration date of the particular prisoner**

```
CREATE TRIGGER add_releaseDate
AFTER INSERT ON caseFiles
FOR EACH ROW
EXECUTE PROCEDURE add_releaseDate();
```

```
INSERT INTO caseFiles ( caseId, incarcerationDate,
                        dateReleased, sentenceLengthYears,
                        isViolentOffense, offense, prid
                      )
  VALUES ( 'c03', '1/1/2014', null, 10, false, 'TESTCRIME', 'p032' );
```

c013	2010-04-28		7	t	manslaughter	p013
c014	2011-03-29		4	f	hacking	p014
c015	2013-01-14		2	t	assault	p015
c03	2014-01-01	2023-12-30	10	f	TESTCRIME	p032

SECURITY

The purpose of this section is to identify and define the user roles associated with this system and then grant or revoke privileges to the various groups

ADMIN

```
CREATE ROLE admin;
GRANT ALL ON ALL TABLES
IN SCHEMA PUBLIC
TO admin;
```

STAFF

```
CREATE ROLE staff;
GRANT SELECT ON prisoners, cells,
                committedInfractions, infractionTypes,
                visitors, visits, prisonAreas,
                prisonerEmployees, staff, people,
                caseFiles, states, guards
TO staff;
GRANT INSERT ON people, staff,
                guards, infirmaryVisits,
                states, caseFiles
TO staff;
GRANT UPDATE ON prisonerEmployees, people, guards, staff, positions
```

GUARDS

```
CREATE ROLE guards;
GRANT SELECT ON prisoners, cells,
                committedInfractions, infractionTypes,
                visitors, visits, prisonAreas,
                prisonerEmployees, staff, people,
                caseFiles, states, guards
TO guards;
GRANT INSERT, UPDATE ON people, prisoners,
                committedInfractions, visitors,
                visits, states, prisonerEmployees
TO guards;
```

NOTES – ISSUES – FUTURE CONSIDERATIONS

If I were to include as much sample data as in a real prison, I would have been able to come up with many more complex queries that would not seem so impressive had they been used on a handful of rows. I have also designated you, Lord Labouseur, the warden.

There are a few issues with the database that I would address with future considerations. The database as is, has no way to address the release of a prisoner, parole possibilities, assigning more than one person to a cell, or a way to prevent violent offenders from being eligible to work as prison employees.

In the limited scope in which we needed to focus, I was forced to omit many aspects of the prison database that would most certainly be necessary in a true implementation. These include but are not limited to, mail deliveries, money account for prisoners, parole systems, cataloging good behavior, accounting for transfers, deaths, etc., organizing shift times, documenting infractions by a guard or complaints against a guard. There is no shortage of potential aspects to account for within a prison institution, and this database represents basic functionality.