HIGH WALL CORRECTIONAL SOLUTIONS

A Database Design Proposal

BY CODY EICHELBERGER
# TABLE OF CONTENTS

- EXECUTIVE SUMMARY ................................................................. 3
- ENTITY RELATIONSHIP DIAGRAM ............................................. 4
- TABLES ..................................................................................... 5
- VIEWS .................................................................................... 18
- REPORTS ............................................................................... 20
- STORED PROCEDURES ......................................................... 23
- TRIGGERS ............................................................................. 27
- SECURITY ............................................................................... 29
- NOTES—ISSUES—FUTURE ...................................................... 31
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  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 \rightarrow firstName, lastName,  
    streetAddress, dateOfBirth, zipCode  

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

### 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
VISITORS lists all visitors

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

functional dependencies
vid \rightarrow 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 \rightarrow heightInches, weightLbs, hairColor, cellId

SAMPLE DATA ON FOLLOWING PAGE
<table>
<thead>
<tr>
<th>prid character(4)</th>
<th>height inches integer</th>
<th>weight lbs integer</th>
<th>haircolor text</th>
<th>cellid integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>p001</td>
<td>67</td>
<td>130</td>
<td>brown</td>
<td>1</td>
</tr>
<tr>
<td>p002</td>
<td>65</td>
<td>120</td>
<td>black</td>
<td>2</td>
</tr>
<tr>
<td>p003</td>
<td>70</td>
<td>144</td>
<td>brown</td>
<td>4</td>
</tr>
<tr>
<td>p004</td>
<td>68</td>
<td>188</td>
<td>blonde</td>
<td>5</td>
</tr>
<tr>
<td>p005</td>
<td>67</td>
<td>133</td>
<td>brown</td>
<td>7</td>
</tr>
<tr>
<td>p006</td>
<td>66</td>
<td>156</td>
<td>red</td>
<td>8</td>
</tr>
<tr>
<td>p007</td>
<td>69</td>
<td>175</td>
<td>blonde</td>
<td>9</td>
</tr>
<tr>
<td>p008</td>
<td>63</td>
<td>129</td>
<td>black</td>
<td>10</td>
</tr>
<tr>
<td>p009</td>
<td>66</td>
<td>150</td>
<td>black</td>
<td>11</td>
</tr>
<tr>
<td>p010</td>
<td>78</td>
<td>175</td>
<td>brown</td>
<td>17</td>
</tr>
<tr>
<td>p011</td>
<td>67</td>
<td>166</td>
<td>brown</td>
<td>12</td>
</tr>
<tr>
<td>p012</td>
<td>59</td>
<td>162</td>
<td>brown</td>
<td>13</td>
</tr>
<tr>
<td>p013</td>
<td>73</td>
<td>143</td>
<td>blonde</td>
<td>14</td>
</tr>
<tr>
<td>p014</td>
<td>75</td>
<td>190</td>
<td>black</td>
<td>15</td>
</tr>
<tr>
<td>p015</td>
<td>69</td>
<td>201</td>
<td>red</td>
<td>16</td>
</tr>
</tbody>
</table>
**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)  
);

**functional dependencies**

prid $\rightarrow$ 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 $\rightarrow$ city, state
**PRISON AREAS**  lists all prison areas

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

*functional dependencies*  
areaId → areaName

**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*  
\text{cellId} \rightarrow \text{cellBlock}
**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 \(\rightarrow\) positionTitle, salaryUSD, areaId, sid  

<table>
<thead>
<tr>
<th>positionId</th>
<th>positionCharacter(4)</th>
<th>positionTitleText</th>
<th>salaryUSD</th>
<th>areaIdCharacter(3)</th>
<th>sidCharacter(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r001</td>
<td>Warden</td>
<td>120000</td>
<td>a01</td>
<td>p020</td>
<td></td>
</tr>
<tr>
<td>r002</td>
<td>Head Doctor</td>
<td>70000</td>
<td>a03</td>
<td>p019</td>
<td></td>
</tr>
<tr>
<td>r003</td>
<td>Head Chef</td>
<td>60000</td>
<td>a04</td>
<td>p018</td>
<td></td>
</tr>
<tr>
<td>r004</td>
<td>Commissary Manager</td>
<td>30000</td>
<td>a10</td>
<td>p017</td>
<td></td>
</tr>
<tr>
<td>r005</td>
<td>Laundry Manager</td>
<td>40000</td>
<td>a13</td>
<td>p016</td>
<td></td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>caseid character(5)</th>
<th>incarceration date</th>
<th>date released date</th>
<th>sentence length years</th>
<th>is violent boolean</th>
<th>offense text</th>
<th>prid character</th>
</tr>
</thead>
<tbody>
<tr>
<td>c001</td>
<td>1999-01-08</td>
<td></td>
<td>20</td>
<td>t</td>
<td>murdered wife</td>
<td>p001</td>
</tr>
<tr>
<td>c002</td>
<td>1994-10-12</td>
<td></td>
<td>25</td>
<td>t</td>
<td>murdered neighbor</td>
<td>p002</td>
</tr>
<tr>
<td>c003</td>
<td>1985-05-14</td>
<td></td>
<td>45</td>
<td>t</td>
<td>rape</td>
<td>p003</td>
</tr>
<tr>
<td>c004</td>
<td>2002-07-02</td>
<td></td>
<td>15</td>
<td>t</td>
<td>armed robbery</td>
<td>p004</td>
</tr>
<tr>
<td>c005</td>
<td>2005-02-03</td>
<td></td>
<td>10</td>
<td>f</td>
<td>drug trafficking</td>
<td>p005</td>
</tr>
<tr>
<td>c006</td>
<td>2013-08-05</td>
<td></td>
<td>10</td>
<td>f</td>
<td>piracy</td>
<td>p006</td>
</tr>
<tr>
<td>c007</td>
<td>2011-11-09</td>
<td></td>
<td>5</td>
<td>f</td>
<td>grand theft</td>
<td>p007</td>
</tr>
<tr>
<td>c008</td>
<td>1992-12-11</td>
<td></td>
<td>20</td>
<td>t</td>
<td>murder</td>
<td>p008</td>
</tr>
<tr>
<td>c009</td>
<td>2014-03-13</td>
<td></td>
<td>8</td>
<td>f</td>
<td>conspiracy to commit murder</td>
<td>p009</td>
</tr>
<tr>
<td>c010</td>
<td>2012-05-15</td>
<td></td>
<td>15</td>
<td>f</td>
<td>drug manufacture</td>
<td>p010</td>
</tr>
<tr>
<td>c011</td>
<td>1980-07-16</td>
<td></td>
<td>150</td>
<td>t</td>
<td>murdered village</td>
<td>p011</td>
</tr>
<tr>
<td>c012</td>
<td>2003-07-22</td>
<td></td>
<td>12</td>
<td>f</td>
<td>tax evasion</td>
<td>p012</td>
</tr>
<tr>
<td>c013</td>
<td>2010-04-28</td>
<td></td>
<td>7</td>
<td>t</td>
<td>manslaughter</td>
<td>p013</td>
</tr>
<tr>
<td>c014</td>
<td>2011-03-29</td>
<td></td>
<td>4</td>
<td>f</td>
<td>hacking</td>
<td>p014</td>
</tr>
<tr>
<td>c015</td>
<td>2013-01-14</td>
<td></td>
<td>2</td>
<td>t</td>
<td>assault</td>
<td>p015</td>
</tr>
</tbody>
</table>
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 \rightarrow infractionName, penalty

<table>
<thead>
<tr>
<th>infractionId</th>
<th>infractionName</th>
<th>penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>i01</td>
<td>Attempted escape</td>
<td>One week of solitary confinement</td>
</tr>
<tr>
<td>i02</td>
<td>Assault on a guard or staff</td>
<td>Two weeks of solitary confinement</td>
</tr>
<tr>
<td>i03</td>
<td>Assault on fellow inmate</td>
<td>Two weeks of solitary confinement</td>
</tr>
<tr>
<td>i04</td>
<td>General insubordination</td>
<td>Revoke yard and commissary privileges</td>
</tr>
<tr>
<td>i05</td>
<td>Inciting a riot</td>
<td>Two days of solitary and no yard privileges</td>
</tr>
<tr>
<td>i06</td>
<td>murder</td>
<td>Transfer to maximum security facility</td>
</tr>
<tr>
<td>i07</td>
<td>possession of contraband</td>
<td>One week of no yard privileges</td>
</tr>
</tbody>
</table>
**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
**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)  
);  

*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)  
);  

*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;

<table>
<thead>
<tr>
<th>firstname text</th>
<th>lastname text</th>
<th>cellnumber integer</th>
<th>cellblockletter character(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joseph</td>
<td>Stalin</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>Ted</td>
<td>Bundy</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>Charles</td>
<td>Manson</td>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>Bernie</td>
<td>Madoff</td>
<td>5</td>
<td>a</td>
</tr>
<tr>
<td>Al</td>
<td>Capone</td>
<td>7</td>
<td>a</td>
</tr>
<tr>
<td>Ted</td>
<td>Kaczynski</td>
<td>8</td>
<td>b</td>
</tr>
<tr>
<td>Adam</td>
<td>Lanza</td>
<td>9</td>
<td>b</td>
</tr>
<tr>
<td>Andrea</td>
<td>Kehoe</td>
<td>10</td>
<td>b</td>
</tr>
<tr>
<td>John</td>
<td>Gacy</td>
<td>11</td>
<td>b</td>
</tr>
<tr>
<td>Jim</td>
<td>jones</td>
<td>12</td>
<td>b</td>
</tr>
<tr>
<td>Scott</td>
<td>Petersor</td>
<td>13</td>
<td>b</td>
</tr>
<tr>
<td>James</td>
<td>Ray</td>
<td>14</td>
<td>c</td>
</tr>
<tr>
<td>Jack</td>
<td>Kevorkic</td>
<td>15</td>
<td>c</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>Dahmer</td>
<td>16</td>
<td>c</td>
</tr>
<tr>
<td>timothy</td>
<td>McVeigh</td>
<td>17</td>
<td>c</td>
</tr>
</tbody>
</table>
**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;

<table>
<thead>
<tr>
<th>position text</th>
<th>firstname text</th>
<th>lastname text</th>
<th>datereleased date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warden</td>
<td>Alan</td>
<td>Labouseur</td>
<td></td>
</tr>
<tr>
<td>Laundry Manager</td>
<td>Adam</td>
<td>Jones</td>
<td></td>
</tr>
<tr>
<td>Head Doctor</td>
<td>Jordon</td>
<td>Aroyo</td>
<td></td>
</tr>
<tr>
<td>Head Chef</td>
<td>Travis</td>
<td>Crabtree</td>
<td></td>
</tr>
<tr>
<td>Commissary Manager</td>
<td>Cody</td>
<td>Eichelberger</td>
<td></td>
</tr>
</tbody>
</table>

**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;

<table>
<thead>
<tr>
<th>firstname text</th>
<th>lastname text</th>
<th>area text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big</td>
<td>Bird</td>
<td>Cafeteria</td>
</tr>
<tr>
<td>Tickle</td>
<td>Elmo</td>
<td>West Yard</td>
</tr>
<tr>
<td>Bobby</td>
<td>Hill</td>
<td>Administration Offices</td>
</tr>
<tr>
<td>Hank</td>
<td>Hill</td>
<td>Infirmary</td>
</tr>
<tr>
<td>Toucan</td>
<td>Sam</td>
<td>East Yard</td>
</tr>
</tbody>
</table>
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
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
```
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
```
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** prisonerVisitors  
this automatically returns a table of the names of visitors that the input prisoner has had

```sql
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;
```
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 $$
DECLARE
    sentenceLength integer := 365 * CAST (new.sentenceLengthYears as INTEGER);
    calcDateReleased TIMESTAMP := new.incarcerationDate + sentenceLength;
BEGIN
    IF NEW.dateReleased IS NULL THEN
        UPDATE caseFiles
        SET dateReleased = calcDateReleased
        WHERE new.prid = prid ;
    END IF;
    RETURN NEW;
END;
$$
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;
**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`();

```sql
INSERT INTO `caseFiles`
VALUES ('c03', '1/1/2014', null, 10, false, 'TESTCRIME', '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.

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

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