--These two functions are used to put each crime into a geographic grid cell based on the latitude and longitude of where a crime occurred.

```
CREATE OR REPLACE FUNCTION x_coordfinder (lon NUMBER)
RETURN NUMBER
IS
v_x NUMBER;
BEGIN
v_x := floor((lon+91.686565684)/0.045);
RETURN(v_x);
END;

CREATE OR REPLACE FUNCTION y_coordfinder (lat NUMBER)
RETURN NUMBER
IS
v_y NUMBER;
BEGIN
v_y := floor((lat-36.619446395)/0.045);
RETURN(v_y);
END;
```

--The following query is used to construct a heat map of crime for the city of Chicago based on the total number of crimes that occurred in each geographic grid cell

```
CREATE OR REPLACE FUNCTION y_coordfinder (lat NUMBER)
RETURN NUMBER
IS
v_y NUMBER;
BEGIN
v_y := floor((lat-36.619446395)/0.045);
RETURN(v_y);
END;
```

--The following PL/SQL is used to construct a crime monitoring system that calculates the critical level of crime that occurs in a geographic grid cell after each new crime that is committed.

--The system will use the following table to allow crimes to be added as they occur in real time

```
Create table crime_tracker (
ID NUMBER NOT NULL,
crime_date timestamp(6) NOT NULL,
arrest varchar2(20),
x number,
```

```
y number,
crime_count number,
critical_level number,
constraint crime_tracker_PK Primary key(ID));

--If a crime results in an arrest, the system will use this trigger to make it count the crime twice

create or replace trigger critical_ranking
before insert on crime_tracker
for each row
begin
if :NEW.arrest = 'true' THEN
:NEW.crime_count := 2;
else
:NEW.crime_count := 1;
END IF;
END;

--For speed, the following table will be used to store the historical daily average of crime for each geographic grid cell. These stored values are created using the following function and PL/SQL code.

Create table historical_averages (
x number,
y number,
historical_average number,
constraint historical_averages_PK Primary key(x,y));

CREATE OR REPLACE FUNCTION arrestcounter (arrest string)
RETURN NUMBER
IS
v_count NUMBER := 0;
BEGIN
IF arrest = 'true'
THEN v_count := 1;
END IF;
RETURN (v_count);
END;

--This query was just used to determine the number of unique days in the data set = 7032
select count(distinct(extract(day from crime_time) || ' ' || extract(month from crime_time) || '
' || extract(year from crime_time))) as num_days
from chicago_crime;
```

```
DECLARE
cursor c_histavg_row is
select x_coordfinder(longitude) as x, y_coordfinder(latitude) as y,
(count(*)+sum(arrestcounter(arrest)))/7032 as histdailyavg
from chicago_crime
where longitude is not null
group by x_coordfinder(longitude), y_coordfinder(latitude);
v_histavgrow c_histavg_row%ROWTYPE;
BEGIN
FOR v_histavgrow in c_histavg_row
LOOP
insert into historical_averages (x,y,historical_average) values
(v_histavgrow.x,v_histavgrow.y,v_histavgrow.histdailyavg);
END LOOP;
END;

--The following function is able to update the critical level of each cell it is called on

CREATE OR REPLACE function crime_counter (a IN NUMBER, b IN NUMBER, c IN TIMESTAMP)
RETURN NUMBER
IS
v_critlevel NUMBER;
v_histavg NUMBER;
v_novalues NUMBER;
BEGIN
select sum(crime_count),count(*) into v_critlevel,v_novalues
from crime_tracker
where crime_tracker.x = a and crime_tracker.y = b
and ((extract(day from c) = extract(day from crime_tracker.crime_date) and extract(hour from
crime_tracker.crime_date) <= extract(hour from c))
or ((extract(day from c) - 1) = extract(day from crime_tracker.crime_date) and extract(hour
from crime_tracker.crime_date) > extract(hour from c)));
select historical_average into v_histavg
from historical_averages
where historical_averages.x = a and historical_averages.y = b;
if v_novalues > 0 THEN
v_critlevel := v_critlevel/v_histavg;
RETURN (round(v_critlevel,2));
ELSE
RETURN(0);
END IF;
END;
```

--The following anonymous PL/SQL is used to upload the crime data in real-time into the system based on a given month and produce the top ten critical grid cells for every hour of every day. The user is prompted to give a specific month from the data set.

```
Declare
v_year int := &year;
v_month int := &monthnumber;
v_days int :=&daysInThisMonth;
Cursor c_crime_row is
select id as id, crime_time as crime_date, arrest as arrest, longitude as lon, latitude as lat
from chicago_crime
where extract(month from crime_time) = v_month and extract(year from crime_time) = v_year
and longitude is not null
order by crime_time;
v_crimerow c_crime_row%ROWTYPE;
v_x NUMBER;
v_y NUMBER;
v_critlevel NUMBER;
v_gridx NUMBER;
v_gridy NUMBER;
v_mostcritical NUMBER;
v_row NUMBER;
Begin
FOR v_crimerow  in c_crime_row
LOOP
v_x := x_coordfinder(v_crimerow.lon);
v_y := y_coordfinder(v_crimerow.lat);
insert into crime_tracker (id,crime_date,arrest,x,y) values
(v_crimerow.id,v_crimerow.crime_date,v_crimerow.arrest,v_x,v_y);
v_critlevel := crime_counter(v_x,v_y,v_crimerow.crime_date);
update crime_tracker set critical_level = v_critlevel where crime_tracker.id = v_crimerow.id;
END LOOP;
FOR i in 1 .. v_days
LOOP
FOR j in 0 .. 23
LOOP
dbms_output.put_line('Day ' || i || ' Hour ' || j || ' Top ten critical levels');
FOR k in 1 .. 10
LOOP
select x,y,mostcritical into v_gridx,v_gridy,v_mostcritical
from (
select x,y,mostcritical,rownum as rn
from (
select x,y,max(critical_level) as mostcritical
```

```
from crime_tracker
where (extract(hour from crime_date) <= j and extract(day from crime_date) = i)
or (extract(day from crime_date) = (i-1) and extract(hour from crime_date) >= j)
group by x,y
order by mostcritical desc))
where rn = k;
dbms_output.put_line(k || ': Cell ' || v_gridx || ' ' || v_gridy || '  Critical Level ' ||
v_mostcritical);
END LOOP;
END LOOP;
END LOOP;
END;
```