This is an analysis of the Los Angeles Police Department crime sheet from 2020 to October 2024. Inspired by the TV show The Rookie, I wanted to see what parts of the city had the most crime and if I could find any trends in criminal activity.

Using this dataset:
https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data

I split my code into three modules: main, plot, and read_data. My main code prints the results, my plot creates the graphs, and my read_data splices the data into the categories I want.

My two goals of the project were to find the areas with the most criminal activity and the most common crimes committed in the LA area. Based on the information I gathered from plotting, I wanted to see if crime activity had any patterns in specific locations.

In my main.rs:

```
mod read_data; //importing the other modules
mod plot;

use std::collections::HashMap;
use chrono::NaiveDate;
use read_data::{read_theft_data, CrimeRecord};
use plot::{plot_crime_over_time, plot_bar_chart};
use std::error::Error;

//counts the number of time x is based on the key
fn aggregate_counts(records: &[CrimeRecord], key_selector: impl Fn(&CrimeRecord) ->
String) -> HashMap<String, usize> {
    let mut counts = HashMap::new();
    for record in records {
        *counts.entry(key_selector(record)).or_insert(0) += 1;
    }
    counts
} // creates a hashmap where a recorded record is given a count for how often it
occurs

fn main() -> Result<(), Box<dyn Error>> {
    let file_path = "Crime_Data_from_2020_to_Present_20241204.csv"; // data path
    let all_data = read_theft_data(file_path)?;
    let specific_crime = "VEHICLE - STOLEN"; //specify the crime & location you want to
plot
```

```
    let specific_location = "Hollywood";

    let filtered_data: Vec<CrimeRecord> = all_data //collect the specified data data
        .iter()
        .filter(|record| record.crime_desc == specific_crime && record.area ==
specific_location)
        .cloned()
        .collect();
        //checks if set is empty
    if !filtered_data.is_empty() {
        let mut crime_counts_over_time = HashMap::new();
        for record in filtered_data {
            if let Ok(date) = NaiveDate::parse_from_str(&record.crime_date, "%m/%d/%Y
%I:%M:%S %p") {
                *crime_counts_over_time.entry(date).or_insert(0) += 1;
            }
        }

        plot_crime_over_time(crime_counts_over_time, "crime_over_time_plot.png")?;
        println!(
            "Crime trend plot for '{}' in '{}' saved as crime_over_time_plot.png.",
            specific_crime, specific_location
        );
    } else {
        println!("No records found for the specified crime and location.");
    }

    let crime_counts = aggregate_counts(&all_data, |record| record.crime_desc.clone());
    plot_bar_chart(
        "top_5_crimes.png",
        "Top 5 Most Common Crimes", //plotting the most common crimes
        crime_counts,
        5, //ask for the number of crimes you want on the graph
    )?;
    println!("Bar chart for the top 5 most common crimes saved as top_5_crimes.png.");


    let location_counts = aggregate_counts(&all_data, |record| record.area.clone());
    plot_bar_chart(
        "top_5_locations.png", //plotting the most common locations where a incident
was reported
        "Top 5 Locations with Most Crimes",
```

```
        location_counts,
        5,
    )?;
    println!("Bar chart for the top 5 locations saved as top_5_locations.png.");

    Ok(())
}

#[cfg(test)]
mod tests {
    use super::*;
    use std::fs::File;
    use std::io::Write;
    use std::env::temp_dir;

    #[test] //testing to see if csv exists, this should pass because i used the dataset
    fn test_missing_file() {
        let file_path = "Crime_Data_from_2020_to_Present_20241204.csv";
        let result = read_theft_data(file_path);
        assert!(result.is_ok(), "expected to fail if the file does not exist");
    }

    #[test] //test to see if the information matches with data template, this should
fail because im testing a incorrect file
    fn test_malformed_data() {
        let mut file_path = std::env::temp_dir();
        file_path.push("malformed_test.csv");
        let mut file = File::create(&file_path).unwrap();
        writeln!(
            file,
            "area,crime_desc,crime_date\n\
            Area1,Theft,2023-01-01\n\
            null row\n\
            Area2,Burglary,2023-01-02"
        )
        .unwrap();
        let result = read_theft_data(file_path.to_str().unwrap());
        assert!(result.is_ok(), "Error for malformed data, but got {:?}", result);
    }
}
```

My plot.rs:

```rust
use plotters::prelude::*;
use chrono::NaiveDate;
use std::collections::HashMap;
use std::error::Error;

pub fn plot_bar_chart(
   output_path: &str,
   title: &str,
   data: HashMap<String, usize>,
   limit: usize,
) -> Result<(), Box<dyn Error>> {
//turns the hashmap into a vector to sort by decending order
   let mut data: Vec<(String, usize)> = data.into_iter().collect();
   data.sort_by(|a, b| b.1.cmp(&a.1));
   let data = data.into_iter().take(limit).collect::<Vec<_>>();
//drawing the plot
   let root = BitMapBackend::new(output_path, (1280, 720)).into_drawing_area();
   root.fill(&WHITE)?;

   let max_count = data.iter().map(|(_, count)| *count).max().unwrap_or(0);
//drawing the bars
   let mut chart = ChartBuilder::on(&root)
       .caption(title, ("sans-serif", 20))
       .x_label_area_size(20)
       .y_label_area_size(50)
       .margin(20)
       .build_cartesian_2d(0..data.len(), 0..max_count)?;
   chart.configure_mesh()
       .x_labels(data.len())
       .y_labels(10)
       .x_label_formatter(&|x| {
           if let Some((label, _)) = data.get(*x) {
               label.clone()
           } else {
               "".to_string()
           }
       })
       .x_desc("Categories")
       .y_desc("Counts")
       .axis_desc_style(("sans-serif", 15))
       .draw()?;
```

```
    chart.draw_series(data.iter().enumerate().map(|(i, (_label, value))| {
        Rectangle::new([(i, 0), (i + 1, *value)], BLUE.filled().stroke_width(1))
    }))?;
    root.present()?;
    Ok(())
}

pub fn plot_crime_over_time( data: HashMap<NaiveDate, usize>, output_path: &str,) ->
Result<(), Box<dyn Error>> {
    let mut sorted_data: Vec<(NaiveDate, usize)> = data.into_iter().collect();
    sorted_data.sort_by_key(|&(date, _)| date);
//sorting the data by data
    let root = BitMapBackend::new(output_path, (1280, 720)).into_drawing_area();
    root.fill(&WHITE)?;
    let max_count = sorted_data.iter().map(|&(_, count)| count).max().unwrap_or(0);

    let (min_date, max_date) = (
        sorted_data.first().map(|&(date, _)| date).unwrap_or_else(||
NaiveDate::from_ymd_opt(1970, 1, 1).unwrap()),
        sorted_data.last().map(|&(date, _)| date).unwrap_or_else(||
NaiveDate::from_ymd_opt(1970, 1, 1).unwrap()),
    );

    let mut chart = ChartBuilder::on(&root)
        .caption("Crime Trends Over Time", ("sans-serif", 20))
        .x_label_area_size(50)
        .y_label_area_size(50)
        .build_cartesian_2d(min_date..max_date, 0..max_count)?;

    chart.configure_mesh().x_labels(10).y_labels(10).draw()?;

    chart.draw_series(LineSeries::new(
        sorted_data.iter().map(|&(date, count)| (date, count)),
        &BLUE,
    ))?
    .label("Crime Count")
    .legend(|(x, y)| PathElement::new(vec![(x, y), (x + 10, y)], &BLUE));

    chart.configure_series_labels().border_style(&BLACK).draw()?;

    root.present()?;
```

```
    Ok(())
}
```

In my read_data.rs:

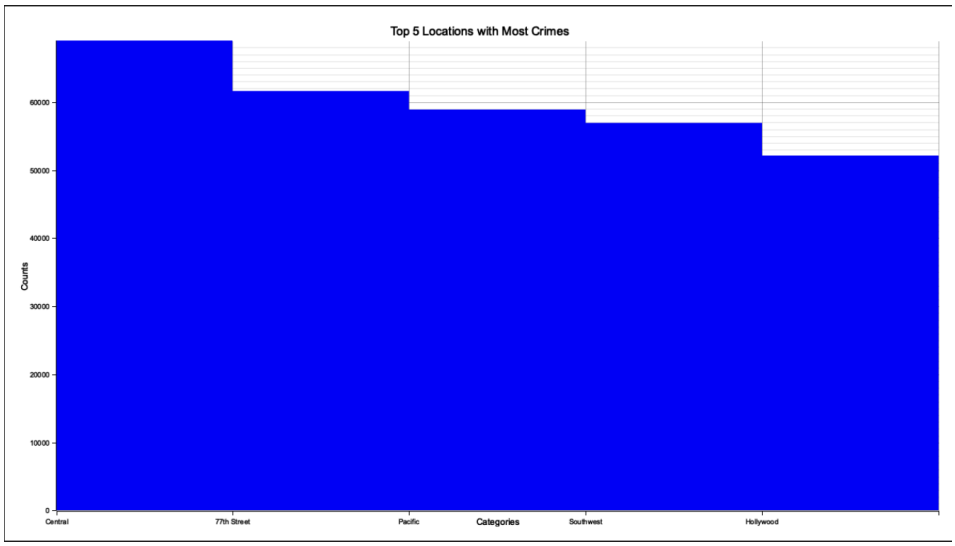
```
use std::error::Error;
use csv::Reader;
use serde::Deserialize;

//pulling the information i need to perform my analysis

#[derive(Debug, Deserialize, Clone)]
pub struct CrimeRecord {
    pub area: String, //location where crime was commited
    pub crime_desc: String, //type of crime
    pub crime_date: String, //date
}
//puts the information from the csv into the struct so i have only the information i
need
pub fn read_theft_data(file_path: &str) -> Result<Vec<CrimeRecord>, Box<dyn Error>> {
    let mut rdr = Reader::from_path(file_path)?;
    let mut theft_locations = Vec::new();

    for result in rdr.records() { //picks what columns have the information i want
        let record = result?;
        let crime_desc = record.get(9).unwrap_or("Unknown").to_string();
        let area_name = record.get(5).unwrap_or("Unknown").to_string();
        let crime_date = record.get(2).unwrap_or("2020-01-01").to_string();

        theft_locations.push(CrimeRecord {
            area: area_name,
            crime_desc,
            crime_date,
        });
    }

    Ok(theft_locations)
}
```

The output should be 3 pngs and this following message:

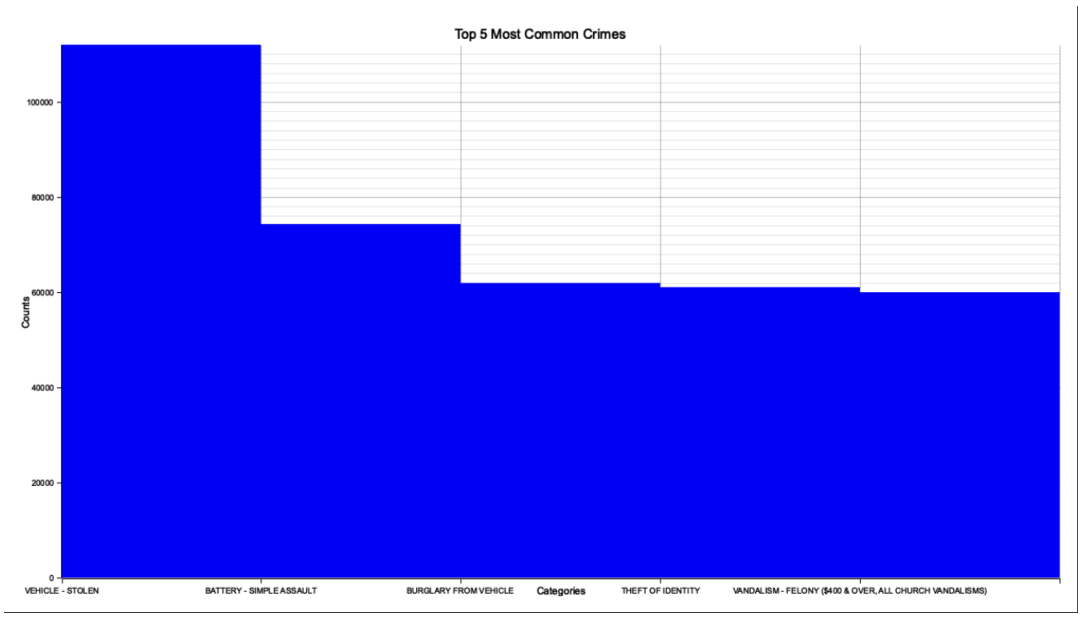
Crime trend plot for 'VEHICLE - STOLEN' in 'Hollywood' saved as crime_over_time_plot.png.

Bar chart for the top 5 most common crimes saved as top_5_crimes.png.
Bar chart for the top 5 locations saved as top_5_locations.png.

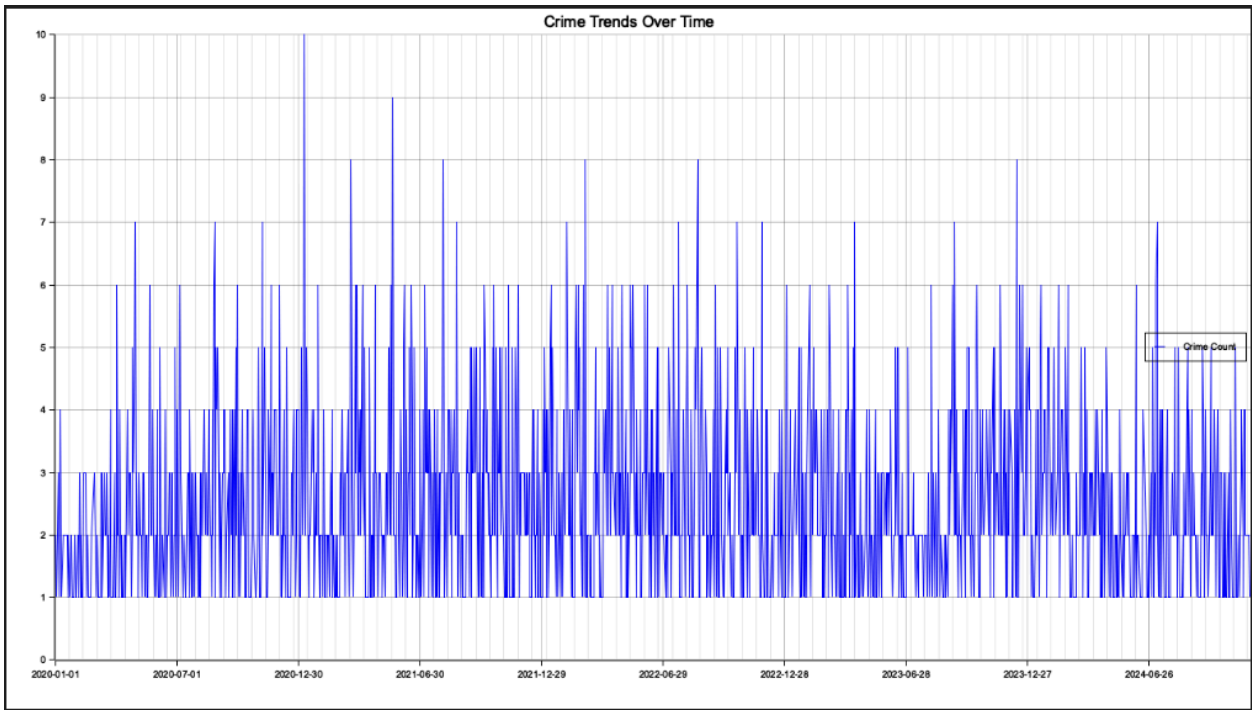
top_5_locations



top_5_crimes.png



crime_over_time_plot.png



My crime over time plot is Hollywood, with vehicles stolen as the crime. I was interested to see if rates have gone down over the years. Based on the plot it seems like there aren't any patterns or trends that show the crime decreasing over the four years data has been collected.

Based on the other information I gathered, I believe that LAPD should invest more resources into central LA and there should be more awareness or precautions to prevent stolen vehicles as it is the most common crime. I found it interesting that burglary from a vehicle was less common than stealing the vehicle itself because I figured you could get away easier with stealing and fleeing the scene rather than getting away with a car. The results of this project were higher than I expected because, in the course of five years, approximately 100000 cars were reported stolen which is absurd. To put that into perspective that's about 55 reports of a car being stolen.