Notebook Technical Information

Data Source

Original Data source is here.

I also uploaded the data I used on Kaggle for anyone to use, here.

Tools Used:

- Python and it’s Libraries,
- Jupyter notebook,

Imports:

The libraries imported for this dataset are following:

- Altair
- Calendar
- Counter from Collections
- NumPy
- Pandas

Purpose

The purpose of this Analysis is to:

1. Find at what time of the day most arrests have happened?
2. Is there any other useful insight that can be found in this data?

Data Cleaning and Processing:

The Original Dataset consists of 19 Columns, with total of 92,865 Rows.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92865 entries, 0 to 92864
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   X                90950 non-null  float64
 1   Y                90950 non-null  float64
 2   OBJECTID         92865 non-null  int64
 3   date_arr         92865 non-null  object
 4   ArrestedAtAddr   92264 non-null  object
 5   case_id          92462 non-null  float64
 6   ar_race          92710 non-null  object
 7   ar_sex           92816 non-null  object
 8   charge           92864 non-null  object
 9   time_arr         92865 non-null  int64
 10  city             92697 non-null  object
 11  state            92784 non-null  object
 12  zip              38620 non-null  object
 13  typebond         83190 non-null  object
 14  bond_amt         92865 non-null  float64
 15  district         90629 non-null  object
 16  tract            90484 non-null  object
 17  zone             87594 non-null  object
 18  reportarea       88922 non-null  object
 19  oau_Arresst_Type  92865 non-null  object
dtypes: float64(4), int64(2), object(14)
memory usage: 14.2+ MB
```

As we can see, many columns are missing values, and many are of wrong datatype. But not All the Columns were needed for my Analysis hence I sliced the data for only the columns that I need. The sliced data had 10 Columns and their details are following.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92865 entries, 0 to 92864
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   OBJECTID        92865 non-null  int64
 1   date_arr        92865 non-null  object
 2   ArrestedAtAddr  92264 non-null  object
 3   case_id         92462 non-null  float64
 4   ar_race         92710 non-null  object
 5   ar_sex          92816 non-null  object
 6   charge          92864 non-null  object
 7   time_arr        92865 non-null  int64
 8   city            92697 non-null  object
 9   state           92784 non-null  object
dtypes: float64(1), int64(2), object(7)
memory usage: 7.1+ MB
```

Finding Problems with Data and their Solution;

- I started by converting the ‘date_arr’ column to it’s right datatype. Then using that column to get the dates of first and last arrest, which are 1999-08-30 and 2023-09-28 respectively.

- Now turning to missing values for now, I used a for loop and a print statement to see how many values in each column are missing, following are the results.

```
0 out of 92865 OBJECTID(s) missing.

0 out of 92865 date_arr(s) missing.

601 out of 92865 ArrestedAtAddr(s) missing.

403 out of 92865 case_id(s) missing.

155 out of 92865 ar_race(s) missing.

49 out of 92865 ar_sex(s) missing.

1 out of 92865 charge(s) missing.

0 out of 92865 time_arr(s) missing.

168 out of 92865 city(s) missing.

81 out of 92865 state(s) missing.
```

The data doesn't have a LOT of missing values, the highest number of missing value columns to lowest are in the following order;

- ArrestedAtAddr
- case_id
- city
- ar_race
- state
- ar_sex
- charge

I opted to replace the Missing Values by 'Missing' in every entry, to not lose any information, dropping the missing values might result in losing valuable data.

- After fixing the missing values, I check the data for any duplicates using df.duplicated(), luckily the dataset has 0 duplicates.

Column Wise Data Check;

I started checking each column of the data for its values and whether there are any problems with information in each column.

1. '**Object Id'**, in my understanding is just an Id for each entry in the original database,
2. '**date_arr'** contains no compromised data,
3. '**ArrestedAtAddr**' has no defined range, but it does contain some missing values that were labelled 'Missing' for ease of understanding,
4. '**case_id**' data is in its min and max limits, although **min case_id was 0** and **max was 9999038314**.
   On further investigation, I found that many Ids had length of 9, but some less than 9 and some had more than 9. To completely understand different lengths, I did more investigation and found that;
   - Only 9 CaseIDs had length of less than 9,
   - 64 CaseIDs had length of 9,
   - 92,389 had length of more than 9.

   On further investigation of IDs with length of greater than 9, I found all of the 92,389 Ids were of length 10. Whilst, length less than 9 had following distribution;

   - 3 IDs of length 4, 3 of length 8, 2 of length 7 and 1 of length 1.

- '**ar_race**', excluding 'Missing' values has, 8 Unique values that were stripped for extra spaces to standardize the data.

```
['B' 'W' 'I' 'A' 'U' 'H' 'O' 'P']
```

- '**ar_sex**', excluding 'Missing' values has, 3 Unique values.

```
['F' 'M' 'U']
```

- ‘**charge**’ column contains a lot of strings that need standardizing that’s why I didn’t explore that in depth, as it would need more NLP techniques to break it down, which at the time of writing this, I don’t possess.
- ‘**time_arr**’, has no missing data, and contains no compromised value.
- ‘**city**’ column contains many duplicates and misspelled version of the same city name, since this data is only about Fayetteville, I fixed all variants of the name Fayetteville.
- ‘**state**’ column contains no inconsistencies, but the Data does contain state name ‘SC’ and ‘AR’. Something to be mindful of when cleaning data.

Data Manipulation:

- Made 3 new columns, ‘**year**’, ’**month_num**’ and ‘**month_name**’,
- Selecting on the data that is for city of “FAYETTEVILLE”,
- Lastly, very little data is available for FAYETTEVILLE before 2010, so I excluded that data because of its very low amount.

Data Analysis;

For ‘time_arr’ data, it doesn’t need any grouping or any other calculation done on it. I move to finding other insights from data;

- Firstly, I grouped ‘year’ and counted OBJECTID columns entries to understand how many Arrests happen per year, named ‘yearly_data’.
- After getting yearly data, I aimed for Average Number of Arrests for each month for all the years. I grouped ‘year’, ‘month_num’ and ‘month_name’ columns and counted entries to get yearly data. Then using that data, grouped ‘month_num’ and ‘month_name’ and averaged the counts for all the years to get the final results, named ‘monthly_mean_data’.
- Next, I sliced the data for Month and time_arr only, to get the patterns of arrest across all the months for all years, named ‘month_time’.
- ‘overall_time_data’ was made by getting all the ‘time_arr’ column and adding ‘time_status’ to it, a column that tells if the time was AM or PM, also calculated yearly average of arrests.
- Next, I aimed for Gender Data Details for all arrests, which was accomplished using groupby and count, and making a new column called ‘percentage’ to calculate percentages.
- Afterwards, I aimed for Gender with Race Data Details for all arrests, which was, again, accomplished by using groupby and count.

Data Visualizations;

All the data visualizations are in the blog article.