# A blue and white background with black text AI-generated content may be incorrect.Advanced Programming and Math

Project: Vanmarcke Aaron

Dataset: [Arrest in LA](https://www.kaggle.com/datasets/arsri1/arrest-data-in-los-angeles/data)

Video:

Contents

[Advanced Programming and Math 1](#_Toc197629123)

[Analyse database: 2](#_Toc197629124)

[Bespreking gekozen zoekopdrachten 2](#_Toc197629125)

[1. Arrestaties per afkomst (query\_arrests\_by\_descent) 2](#_Toc197629126)

[2. Arrestaties per gebied (query\_arrests\_by\_area) 2](#_Toc197629127)

[3. Leeftijdsverdeling (query\_age\_distribution) 3](#_Toc197629128)

[4. Meest voorkomende misdrijf (query\_most\_common\_crime) 3](#_Toc197629129)

[Schema Applicatie 4](#_Toc197629130)

[Geïmplementeerde Delen 5](#_Toc197629131)

[Moeilijkheden 6](#_Toc197629132)

## Analyse database:

**Layout**:

* Nominale variabelen: Report Type, Area Name, Sex Code, Descent Code, Disposition  
  Description, Address, Cross Street, Booking Location.
* Ordinale variabelen: Geen duidelijke ordinale variabelen in deze dataset.
* Discrete variabelen: Report ID, Area ID, Reporting District, Booking Location Code.
* Continue variabelen: Arrest Date, Time, Age, LAT, LON, Location, Booking Date, Booking  
  Time.

**Centrummaat dataset**:

Gemiddeld zijn ze 35 jaar oud, en is het meestal een Booking, en is het meestal een man.

**Dataset preprocessing** :

1. **Omzetten van datums naar Datetime objecten**: Voor eenvoudige tijdsanalyses en het genereren van tijdsgebonden plots.
2. **Verwijderen van ontbrekende waarden**: Om ervoor te zorgen dat de dataset volledig en betrouwbaar is voor analyses.
3. **Encoderen van categorische variabelen**: Voor betere prestaties en eenvoudige verwerking van categorische gegevens.
4. **Verwijderen van outliers**: Om de dataset representatief te houden en foutieve waarden te elimineren.

Heatmap:

A chart with red and blue squares

AI-generated content may be incorrect.

## Bespreking gekozen zoekopdrachten

### 1. Arrestaties per afkomst (query\_arrests\_by\_descent)

**Wat doet het?**  
Laat zien hoeveel mensen zijn gearresteerd per afkomstscode (bijv. "H" voor Hindustani).

**Waarom handig?**

* Je ziet arrestatietrends per etnische groep.
* Handig voor onderzoeken naar ongelijkheid.
* Je kunt zelf een code kiezen.

**Voorbeeld:**  
Invoeren van "H" geeft: **155.860 arrestaties** voor Hindustani.

### 2. Arrestaties per gebied (query\_arrests\_by\_area)

**Wat doet het?**  
Geeft het aantal arrestaties in een bepaald gebied op basis van Area ID (bijv. "5").

**Waarom handig?**

* Je ziet waar veel arrestaties plaatsvinden.
* Politie kan hiermee beter plannen.
* Je kunt trends per gebied volgen.

**Voorbeeld:**  
Invoeren van "5" geeft: **20.345 arrestaties** in gebied 5.

### 3. Leeftijdsverdeling (query\_age\_distribution)

**Wat doet het?**  
Toont in een grafiek hoeveel arrestaties er per leeftijdsgroep zijn.

**Waarom handig?**

* Je ziet welke leeftijdsgroepen vaak worden gearresteerd.
* Maatregelen kunnen beter worden gericht.
* Makkelijk te begrijpen door de grafiek.

**Voorbeeld:**  
Meeste arrestaties zijn bij mensen tussen **23,8 en 32,2 jaar**.

### 4. Meest voorkomende misdrijf (query\_most\_common\_crime)

**Wat doet het?**  
Laat het vaakst voorkomende misdrijf zien, met het aantal keer dat het voorkomt.

**Waarom handig?**

* Laat zien wat de grootste problemen zijn.
* Politie kan hierop focussen.
* Je kunt filteren op misdrijftypes.

**Voorbeeld:**  
Diefstal komt het meest voor: **50.321 keer**.

## Schema Applicatie

Client-Side (Frontend)

├── GUI (Tkinter of andere GUI-bibliotheek)

│ ├── Gebruikersinteractie (knoppen, invoervelden)

│ ├── Visualisatie van data (plots, tabellen)

│ └── De GUI’s

└── Client/Logic

├── communication.py

│ ├── Verantwoordelijk voor netwerkcommunicatie

│ ├── Methoden zoals send(), receive\_json(), query\_json\_receive()

│ └── Luistert naar serverberichten

└── queries.py

├── Functies voor specifieke queries (bijv. query\_arrests\_by\_area)

└── Verwerkt serverantwoorden en retourneert resultaten aan de GUI

Server-Side (Backend)

├── Core Functionaliteit

│ ├── dispatcher.py

│ │ ├── Verwerkt inkomende acties van de client

│ │ └── Roept specifieke logica aan (bijv. queries of plots)

│ └── logic/

│ ├── plots.py

│ │ ├── Preprocessing van data

│ │ ├── Genereren van visualisaties en samenvattingen

│ │ └── Retourneert data in JSON-formaat

│ └── queries.py

│ ├── Database-interacties

│ └── Ophalen van specifieke datasets

└── Database

├── Arrest\_Data\_from\_2020\_to\_Present.csv

└── Data-opslag voor analyses en visualisaties

## Geïmplementeerde Delen

Server:

- belangrijke acties via logvenster kunnen bekijken  
 - overzicht van de ingelogde cliënts  
 - de gegevens van de cliënts opvragen  
 - per client overzicht tonen van de opgevraagde zoekopdrachten  
 - bericht aan de aangemelde cliënt(s) kunnen opsturen  
 - populariteit van de aangeboden zoekopdrachten kunnen bekijken  
 - server kunnen afsluiten  
 **Langs cliëntzijde dient de gebruiker (logischer wijs):**

- kunnen in- en uitloggen;  
 - zoekopdrachten (met parameters) doorsturen  
 - resultaten kunnen weergeven  
 - berichten van de server kunnen weergeven

**Opmerkingen:**  
 - client moet zich registreren (naam, nickname, emailadres) en/of aanmelden;  
 - de moderator kan ten alle tijde kunnen nagaan wie er online is;  
 - de (minimale) gegevens van een cliënt (naam, nickname, emailadres) worden bewaard;  
 - de moderator moet de gegevens van alle cliënts kunnen opvragen;  
 - de client moet verschillende zoekopdrachten kunnen opvragen. Waar nodig worden ook  
extra parameters doorgestuurd. De zoekopdrachten gebeuren langs serverzijde op de  
beschikbare data.  
 - De cliënt heeft géén toegang tot de dataset.  
 - de cliënt geeft de gegevens op via tabel of een grafische manier (afbeelding) weer.  
 - de servermoderator houdt bij welke zoekopdrachten er opgevraagd worden.  
 - de servermoderator kan ook raadplegen welke zoekopdrachten er populair zijn.  
 - de servermoderator kan een melding naar de aangemelde cliënts sturen.

## Moeilijkheden

**1. Broadcast thread werkte niet goed**  
Ik had moeite met het actief houden van een thread die luistert naar serverberichten. Soms crashte het als de server ineens stopte.  
**Oplossing:**  
Ik heb een aparte thread gemaakt met threading en foutafhandeling toegevoegd (try-except). Zo blijft de thread draaien, ook bij fouten.

**2. Plots versturen ging fout**  
Ik probeerde eerst afbeeldingen van de server naar de client te sturen, maar dat gaf veel problemen.  
**Oplossing:**  
In plaats van afbeeldingen stuur ik nu de plotdata als JSON. Dat is makkelijker te verzenden, ontvangen en weer te geven.

Had vervolgens ook heel wat problemen met het decoderen van deze json data.