**RD - Desenvolvedor Business Intelligence - Projeto Desafio**

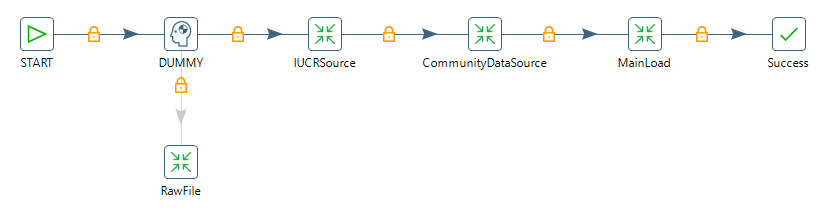
Guilherme Ocker Ribeiro

**ETL e definição do modelo ER:**

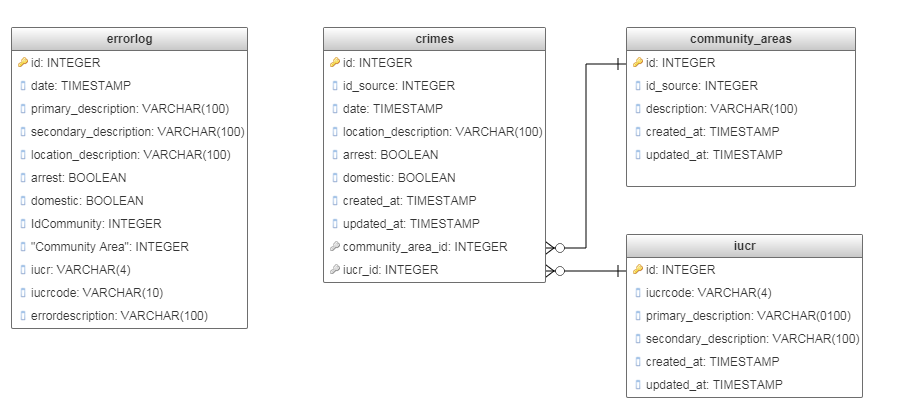
Foi utilizada a ferramenta PDI - Kettle, da suite Pentaho. Ferramenta bem reconhecida no mercado, agnóstica quanto a fontes e destinos e se trata de uma ferramenta onde possuo experiência de alguns anos. ([Download](http://community.pentaho.com/projects/data-integration/))

Para facilitar o início do desenvolvimento do processo de ETL, foi extraído uma amostra da carga completa, com algumas milhares de linhas, facilitando o processo de testes iniciais, construção do modelo e debug, além da análise completa do arquivo através de uma exportação do arquivo “cru” para a base de dados, possibilitando visualizar a natureza dos dados e seus relacionamentos.

**MainJob:**



**Modelo ER:**



A tabela *Community Areas* foi extraída a partir deste [link](https://data.cityofchicago.org/Facilities-Geographic-Boundaries/Boundaries-Community-Areas-current-/cauq-8yn6) e importado para dentro do modelo a partir de um ETL específico. A tabela IUCR (*Illinois Uniform Crime Reporting*) foi extraída deste [link](https://data.cityofchicago.org/Public-Safety/Chicago-Police-Department-Illinois-Uniform-Crime-R/c7ck-438e) e também tratada e importada para dentro do modelo através de ETL específico.

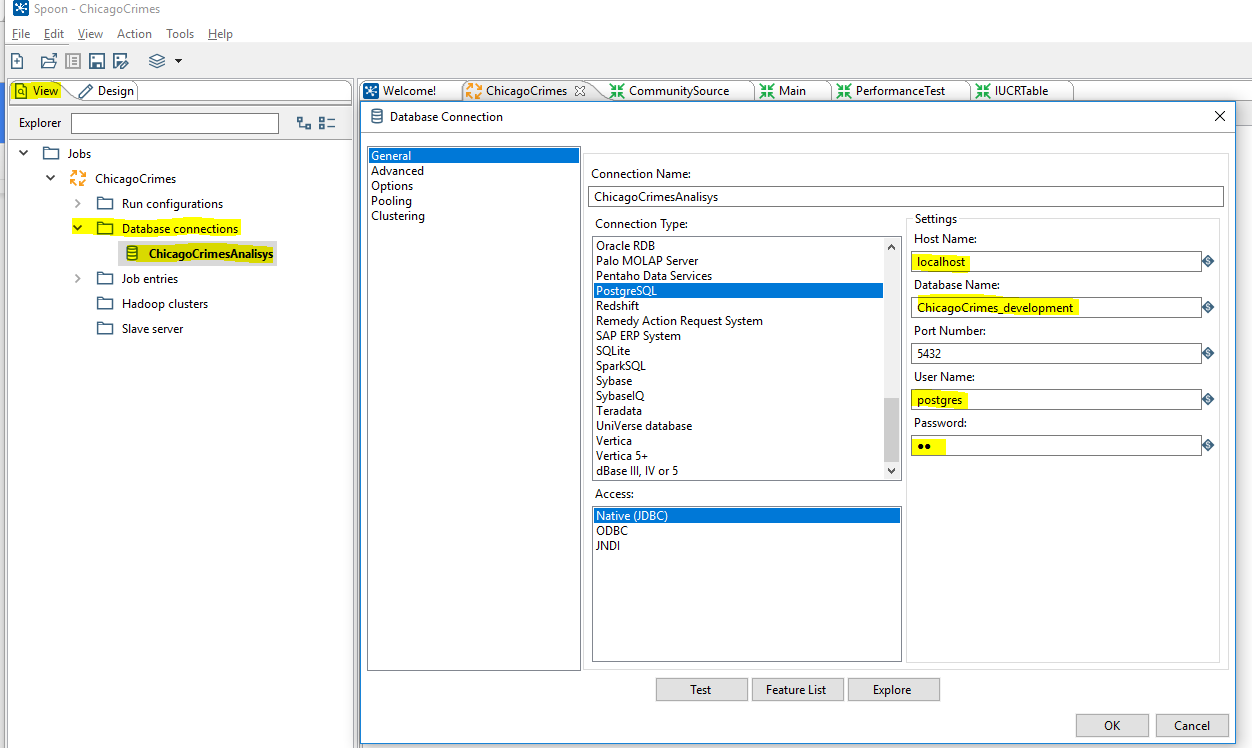
**Configurações e Execução do ETL**

[Download](http://community.pentaho.com/projects/data-integration/) - Executar ~\pdi-ce-7.1.0.0-12\data-integration\**Spoon.bat**

Se não abrir, pode ser necessário alterar as configurações de criação da JVM, através dos seguintes passos:

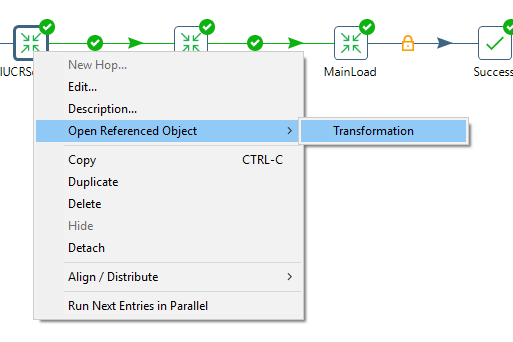
* Editar o arquivo Spoon.bat, próximo a linha 94, identificar e alterar para os seguintes parâmetros: ***PENTAHO\_DI\_JAVA\_OPTIONS="-Xms512m" "-Xmx1024m" "-XX:MaxPermSize=256m"***
* Executar novamente o arquivo Spoon.bat.

**Configuração de banco:**



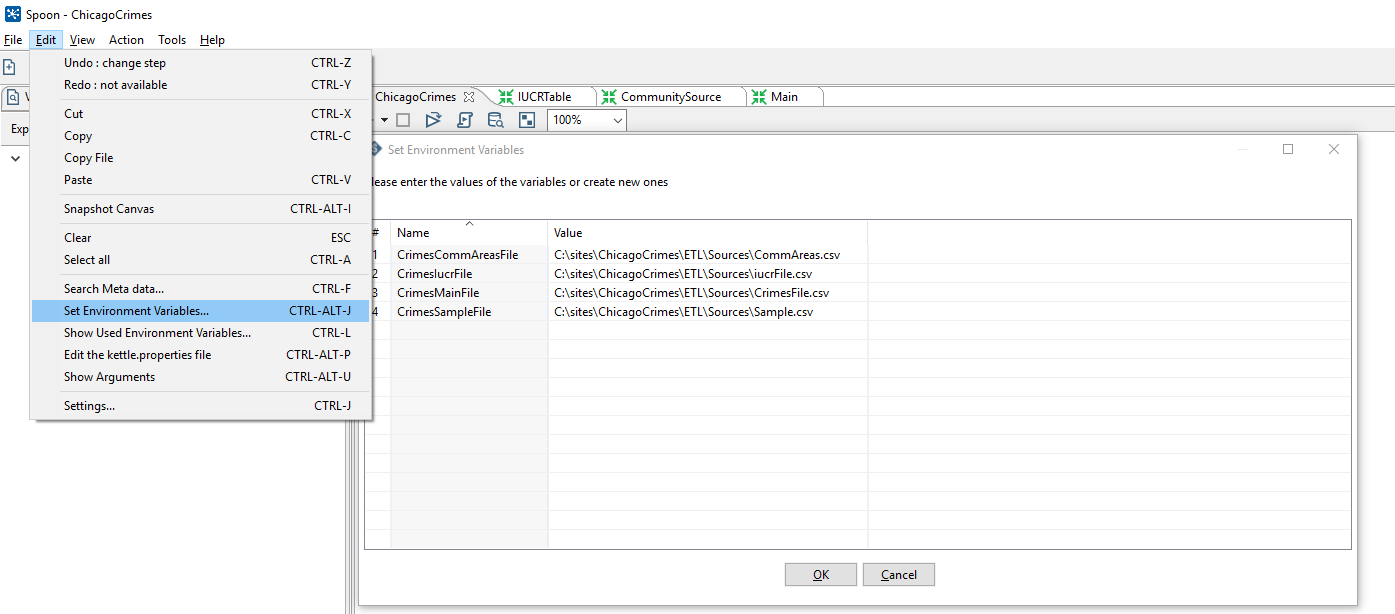
**Arquivo de Job:**

Abrir o arquivo de *job ChicagoCrimes.kjb* dentro da pasta de ETL do projeto. A partir dele, é possível “navegar” até os outros arquivos *transformation* se for de interesse:



**Variaveis de ambiente Kettle:**

Serão utilizadas variáveis de ambiente para facilitar a identificação dos arquivos de fonte. Configurar de acordo com o print e valores da tabela abaixo.



O arquivo *CrimesFile.csv* deve ser baixado através deste [link](https://data.cityofchicago.org/api/views/ijzp-q8t2/rows.csv?accessType=DOWNLOAD), renomeado e colocado na pasta Sources, como mostrado abaixo.

|  |  |
| --- | --- |
| CrimesIucrFile | C:\sites\ChicagoCrimes\ETL\Sources\iucrFile.csv |
| CrimesCommAreasFile | C:\sites\ChicagoCrimes\ETL\Sources\CommAreas.csv |
| CrimesMainFile | C:\sites\ChicagoCrimes\ETL\Sources\CrimesFile.csv |
| CrimesSampleFile | C:\sites\ChicagoCrimes\ETL\Sources\Sample.csv |

**Execução:**

A execução se dá de maneira completa através do job *ChicagoCrimes.kjb*. No “*Main File Input*” da transformation *main.ktr*, pode-se escolher entre o arquivo de amostra e o arquivo principal, utilizando as variáveis acima. Para uma nova execução do job completo, é importante que as tabelas estejam ***truncadas****.* Pode-se executar usando a tecla F9 ou clicando no botão “*Run*” e em seguida “Run” novamente.

Execuções faseadas podem ser feitas acessando cada *transformation*.

**Em caso de erro**, analisar rapidamente as linhas vermelhas no painel Logging, principais erros são:

* Arquivo não encontrado -> Verificar pasta source, nomenclatura dos arquivos e se as variáveis setadas estão de acordo com o caminho e nomes dos arquivos.
* Conexão com BD -> Verificar se a conexão ao BD está correta de acordo com o print acima, certificar-se de que a conexão esteja acessível em todas as *transformations* (*Shared connection* deve estar em negrito, se não, “*Menu de contexto*” >> “*Share*”).
* Tabelas destino não truncadas -> Executar *Truncate table* nas respectivas tabelas.

**Performance do ETL:**

A execução do arquivo de amostra leva alguns segundos e do arquivo principal leva cerca de 9 minutos.

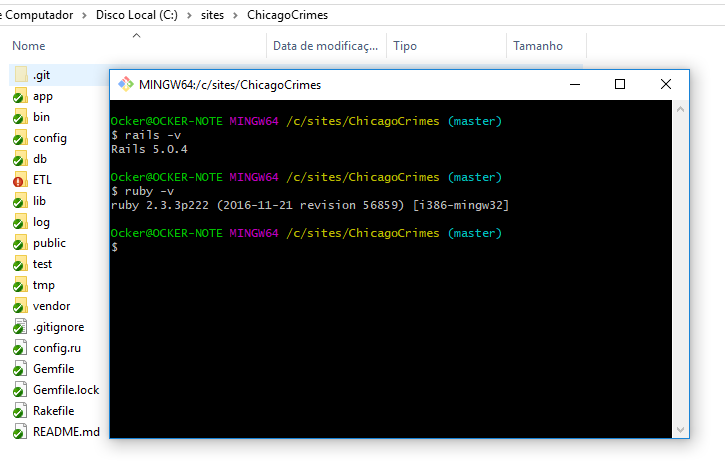
Nas primeiras versões da modelagem, com a base relacional sendo criada “na mão”, o Job principal levava cerca de 3 minutos para ser executado. Porém, a base relacional teve que ser criada novamente a partir do uso do RoR, fazendo com que a performance fosse degradada por algum motivo.

Não houve tempo hábil para estudar e corrigir essa degradação, mas o ganho de performance final na execução é bastante provável.

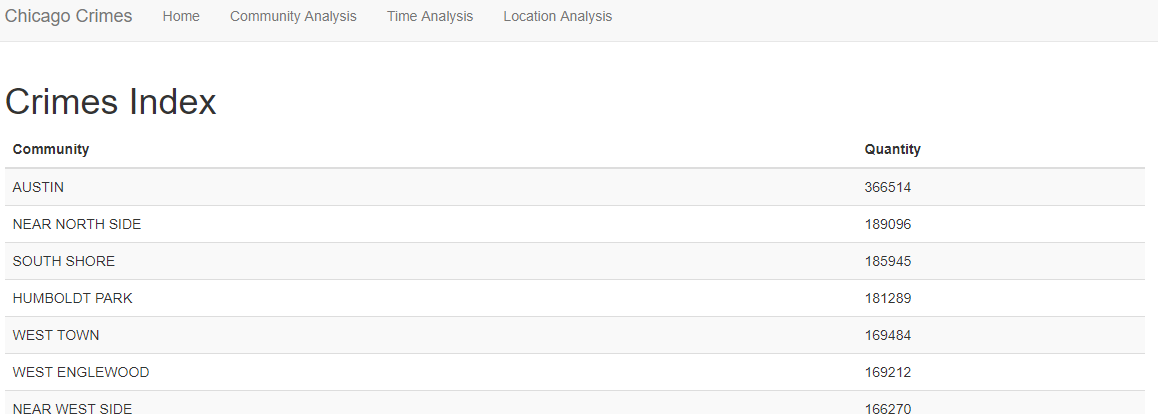
**Aplicação ChicagoCrimesAnalysis:**

**GIT:** <https://github.com/BI-RDChicago/ChicagoCrimesAnalysis>

**Versões do Ruby e Rails:**



**Aplicação:**

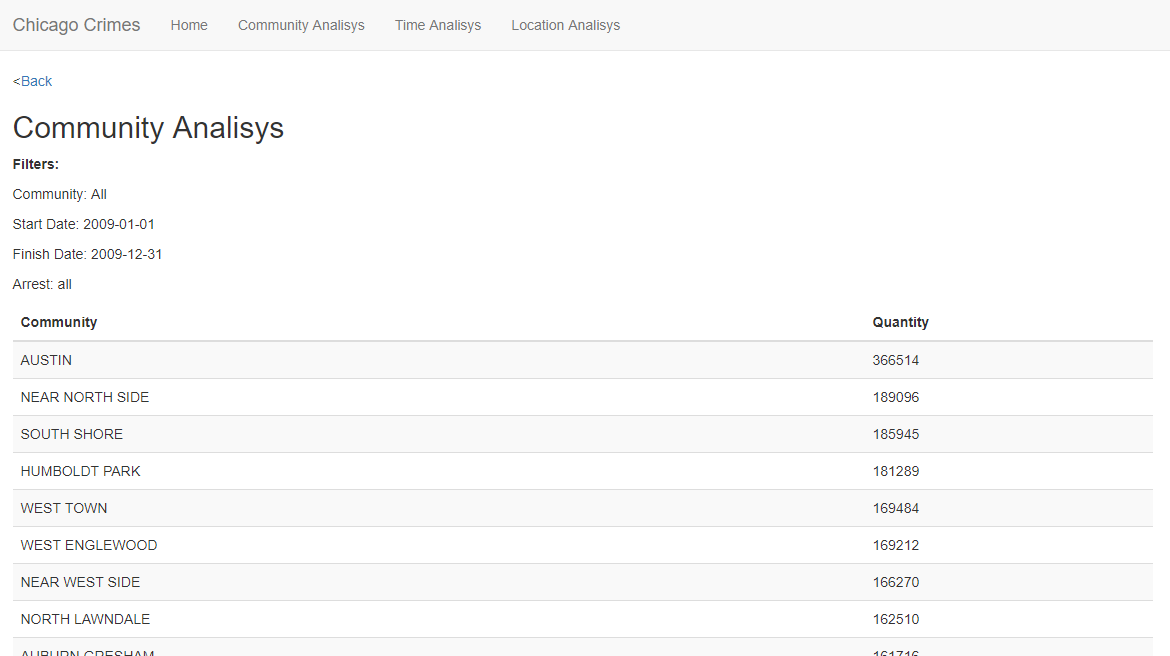


Foram criadas 3 análises básicas para compor o desafio proposto:

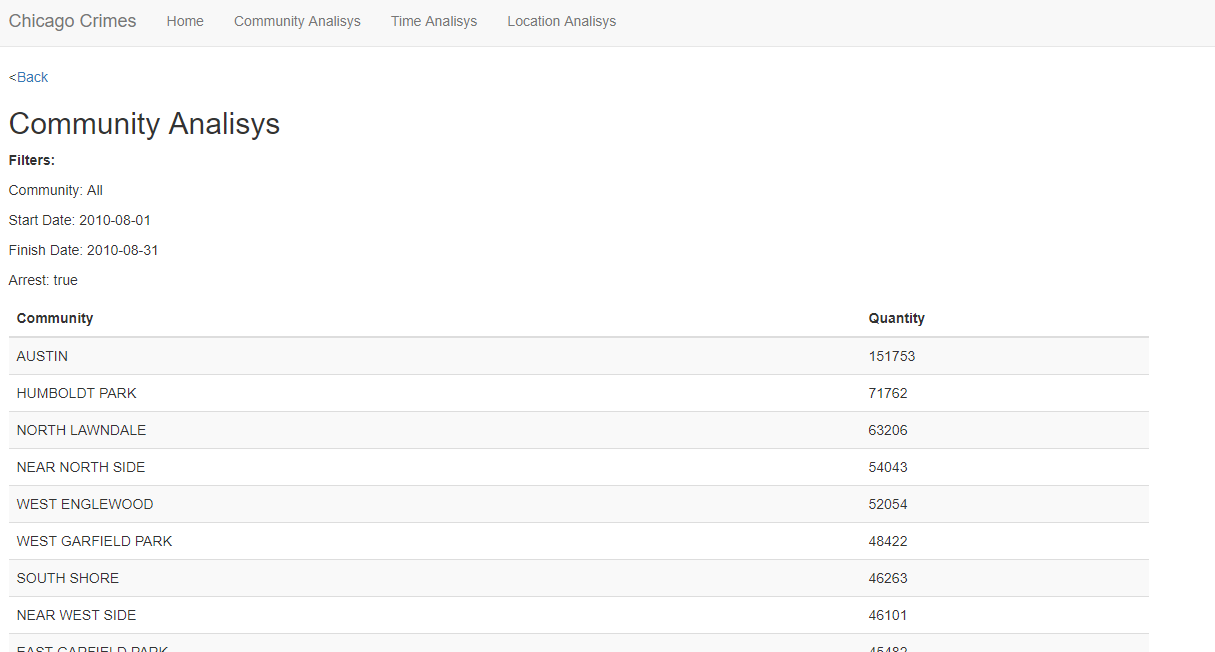
* Community Analysis - Agrupado por Community
* Time Analysis - Agrupado por tempo
* Location Analysis - Agrupado por local do crime

**Perguntas do desafio:**

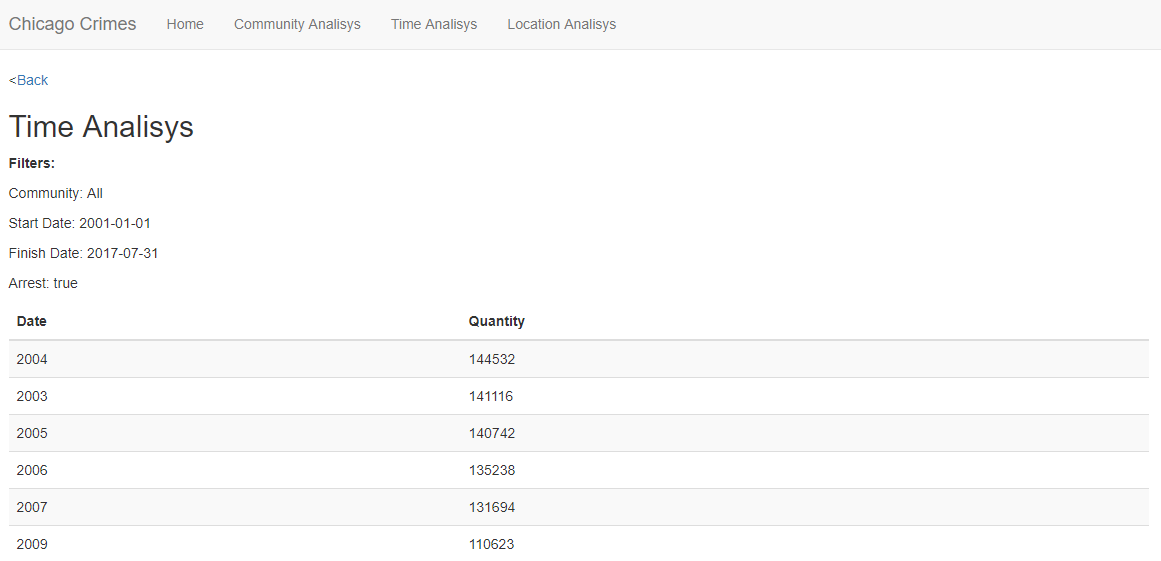
1. Qual **Community Area** teve mais crimes no ano de 2009:



1. Qual **Community Area** gerou mais prisões no mês de Agosto de 2010:

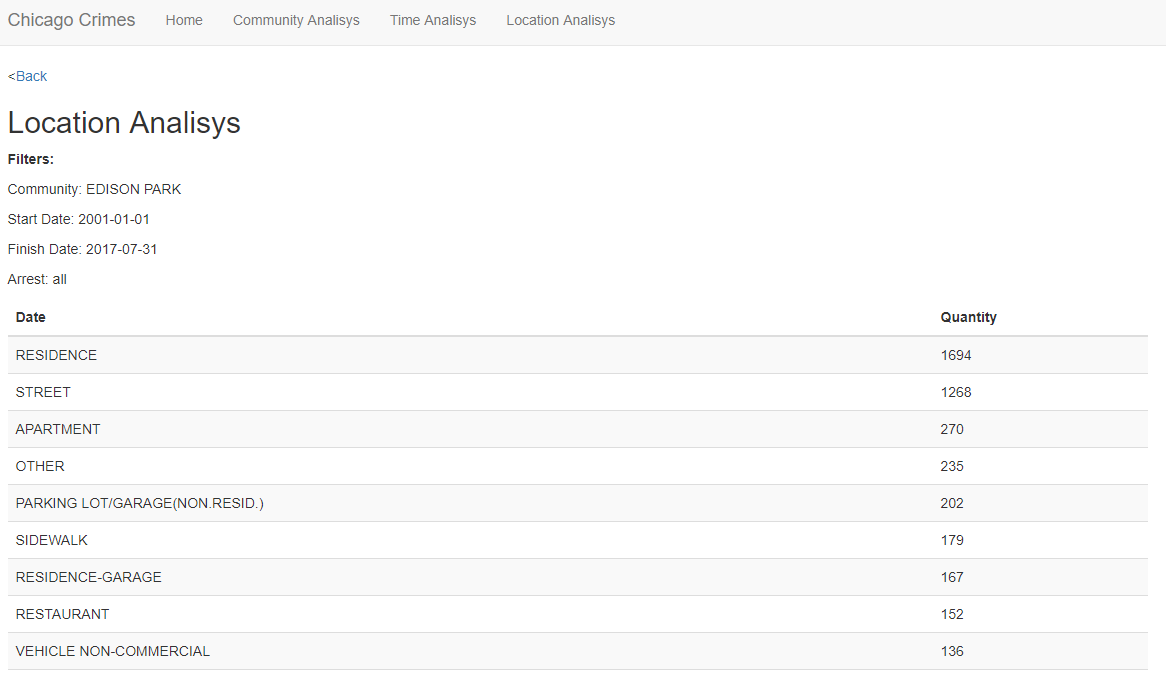


1. Qual foi o ano com mais prisões em Chicago?



1. Crie pelo menos mais uma pergunta que deverá ser respondida pelo seu DW.

* Qual a distribuição, por localização do crime, para **Community Area** com menos crimes no histórico?



1. Fique livre para criar outras análises que você achar interessante.

O item “Location analysis” permite agrupar por local do crime, permitindo ver a distribuição dos crimes nos locais, filtrando por “Community Areas” mais ou menos “perigosas”.

**Detalhes do desenvolvimento e melhorias sugeridas:**

A parte de modelagem e ETL ficaram usáveis com poucas horas de trabalho, sendo dedicando mais tempo a controle de erros, validações e performance. Porém, ao começar o estudo de RoR, acabei percebendo que ligar uma aplicação RoR a uma base “legada/não nativa” era muito mais custoso do que recriar a modelagem através das linhas de comando (Git Bash) devido à falta de conhecimento e alto acoplamento entre RoR e a base.

Infelizmente, por algum motivo, a performance do ETL sofreu uma degradação considerável após a utilização da nova modelagem. Não houve tempo para estudar o porquê disso, mas creio que seja possível reduzir o tempo total do ETL, hoje próximo aos 9 min, para os 3 min que havia encontrado com a modelagem desacoplada ao RoR.

Apesar de ter normalizado os dados de crimes, com o uso de tabelas relacionais, *IUCR*, *CommunityArea* e podendo normalizar ainda uma tabela *Location*, a utilização (DW ou relacional?) e performance podem ditar a futura escalabilidade, forçando a “denormalização” do modelo, aproximando-o a um modelo de DW, com a tabela de crimes sendo “Fato”, agregando os valores relevantes das outras tabelas, sem uso de Joins, gerando dimensões de apoio.

Diria que 70% do tempo dedicado ao desafio foi consumido na criação da aplicação e estudo do Rails.

As ferramentas básicas utilizadas para a aplicação: Git Bash e Sublime Text.

As múltiplas páginas da aplicação, communityfilter.html.erb e communityfilterapply.html.erb por exemplo, existem por não ter conseguido trabalhar com a atualização da tabela na própria página. Precisaria dedicar um pouco mais de tempo lendo sobre Partials pages, Selective Refresh ou até mesmo ajax + JS para permitir esse tipo de ação e melhorar

Na Crimes\_controler, foram utilizadas técnicas básicas de query dinâmica, com *scope*, para evitar repetições de código, porém, creio que exista maneiras melhores de serem feitas. Além de achar que parte dessas consultas seriam inerentes a model, e não à controller.

**Repositório:**

**Git – Branch:**

<https://github.com/BI-RDChicago/ChicagoCrimesAnalysis/tree/Entrega1_Branch>