Korzystanie z Big Data z Google Cloud

Do tej pory mieliśmy do czynienia ze stosunkowo niewielkimi zbiorami danych. Teraz użyjemy dużych zbiorów danych — w niektórych przypadkach bardzo dużych zbiorów danych, które zmieniają się co godzinę! Czasami pracujemy na wystarczająco mocnym komputerze, aby móc pobrać taki duży zestaw danych, ale nie każdy duży zestaw danych można pobrać. Jeśli o to chodzi, nie wszystkie zestawy danych można legalnie pobrać. A w przypadku bazy danych jakości powietrza trzeba by co godzinę pobierać nową wersję. W takich przypadkach lepiej zostawić dane tam, gdzie są i użyć chmury do wykonania pracy. Interesującą konsekwencją ciężkiej pracy w chmurze jest to, że komputer nie musi być bardzo duży ani szybki. Po prostu pozwól chmurze wykonać pracę związaną z bazą danych i analizą.

Co to jest Big Data?

Big data odnosi się do zbiorów danych, które są zbyt duże lub złożone, aby można było nimi zarządzać przy użyciu tradycyjnych technik przetwarzania danych. Dane z wieloma obserwacjami i wieloma wierszami zapewniają większą dostępność wyrafinowanych technik statystycznych i generalnie prowadzą do mniejszego wskaźnika fałszywych odkryć. Jak omówiliśmy w rozdziale 1 tego minibooka, duże zbiory danych stają się coraz bardziej powszechne w naszym społeczeństwie, ponieważ liczba komputerów i czujników rośnie i tworzy coraz więcej danych w coraz szybszym tempie. W tej części mówimy o korzystaniu z chmury w celu uzyskania dostępu do tych dużych baz danych za pomocą Pythona i Pand, a następnie wizualizacji wyników na Raspberry Pi.

Zrozumienie Google Cloud i BigQuery

Cóż, czasami, aby uzyskać dostęp do Big Data, musisz użyć BigQuery. Ważne jest, aby zrozumieć, że nie tylko przechowujesz dane w chmurze, ale także korzystasz z narzędzi do analizy danych w chmurze. Zasadniczo używasz swojego komputera do dowodzenia tym, co te komputery w chmurze robią z danymi.

Platforma Google Cloud

Google Cloud Platform to pakiet usług przetwarzania w chmurze, które działają w tej samej infrastrukturze, co produkty Google dla użytkowników końcowych, takie jak wyszukiwarka Google i YouTube. Jest to strategia chmurowa, która została z powodzeniem zastosowana w firmach Amazon i Microsoft. Wydaje się, że korzystanie z własnych usług i produktów związanych z danymi w celu zbudowania oferty chmurowej naprawdę tworzy dobre środowisko zarówno dla użytkownika, jak i dla firmy, umożliwiające czerpanie korzyści z postępów i ulepszeń zarówno produktów, jak i chmur. Platforma Google Cloud zawiera ponad 100 różnych interfejsów API (interfejsów programowania aplikacji) i produktów usług danych dostępnych dla nauki o danych i sztucznej inteligencji. Podstawową usługą, z której korzystamy w tym rozdziale, jest Google API o nazwie BigQuery.

BigQuery od Google

System oprogramowania REST (Representational State Transfer) to zestaw kodu, który definiuje zestaw struktur komunikacyjnych używanych do tworzenia usług sieciowych, zwykle wykorzystujących do komunikacji żądania http i https. Zapewnia to duży zestaw interoperacyjności dla różnych komputerów z różnymi systemami operacyjnymi, które próbują uzyskać dostęp do tej samej usługi sieciowej. BigQuery opiera się na usłudze internetowej RESTful (pomyśl o kontaktowaniu się ze stronami internetowymi z adresami URL, które zadają określone pytania w standardowym formacie, a następnie otrzymywaniu odpowiedzi z powrotem, tak jak przeglądarka otrzymuje stronę internetową) oraz wielu bibliotek dla Pythona i innych języków ukrytych złożoność zapytań przechodzących tam iz powrotem. Abstrakcja w systemach oprogramowania jest kluczem do tego, aby duże systemy działały

i były rozsądne w programowaniu. Na przykład, chociaż przeglądarka internetowa używa HTML do wyświetlania stron internetowych, istnieją pod nią warstwy oprogramowania, wykonujące takie czynności, jak przesyłanie pakietów IP lub manipulowanie bitami. Te niższe warstwy różnią się, jeśli korzystasz z sieci przewodowej lub sieci Wi-Fi. Fajną rzeczą w abstrakcji jest tutaj poziom strony internetowej, nie obchodzi nas to. Po prostu go używamy. BigQuery to model bezserwerowy. Oznacza to, że BigQuery ma najwyższy poziom abstrakcji w społeczności chmurowej, usuwając z użytkownika odpowiedzialność za martwienie się o uruchamianie maszyn wirtualnych (wprowadzanie nowych maszyn wirtualnych online w chmurze), pamięć RAM, liczbę procesorów i tak dalej. Możesz skalować od jednego do tysięcy procesorów w ciągu kilku sekund, płacąc tylko za faktycznie używane zasoby. Zrozum, że w tej książce Google pozwoli ci korzystać z chmury za darmo, więc nie będziesz musiał w ogóle płacić podczas okresu próbnego. BigQuery ma dużą liczbę publicznych zbiorów danych big data, takich jak Medicare i NOAA (National Oceanic and Atmospheric Agency). Korzystamy z tych zestawów danych w poniższych przykładach. Jedną z najciekawszych funkcji BigQuery jest możliwość strumieniowego przesyłania danych do BigQuery rzędu milionów wierszy (próbek danych) na sekundę, czyli danych, które możesz zacząć analizować niemal natychmiast. Będziemy używać BigQuery z pandas biblioteki Pythona. Biblioteka Pythona google.cloud udostępnia bibliotekę Pythona, która odwzorowuje dane BigQuery na nasze przyjazne pandy DataFrames znane z części 2.

Bezpieczeństwo komputera w chmurze

Bylibyśmy niedbali, gdybyśmy nie porozmawiali trochę o zachowaniu dobrego bezpieczeństwa komputera podczas korzystania z chmury. Google osiąga to, stosując paradygmat IAM (zarządzanie tożsamością i dostępem) we wszystkich swoich ofertach chmurowych. Dzięki temu administratorzy mogą autoryzować, kto może podejmować jakie działania na określonych zasobach, zapewniając pełną kontrolę i widoczność prostych projektów, a także precyzyjny dostęp rozciągający się na całe przedsiębiorstwo. W poniższych sekcjach pokażemy, jak skonfigurować uwierzytelnianie IAM.

PUBLICZNEJ BAZY MEDICARE

Medicare to narodowy program ubezpieczeń zdrowotnych (pojedynczy płatnik) w Stanach Zjednoczonych, zarządzany przez Centers for Medicare and Medicade Services (CMS). Zapewnia ubezpieczenie zdrowotne dla Amerykanów w wieku 65 lat i starszych. Zapewnia również ubezpieczenie zdrowotne młodszym osobom z pewnymi niepełnosprawnościami i schorzeniami. W 2017 roku zapewnił ubezpieczenie zdrowotne ponad 58 milionom osób. Z 58 milionami osób w systemie, Medicare generuje każdego roku ogromną ilość dużych zbiorów danych. Google i CMS połączyły siły, aby umieścić dużą ilość tych danych w publicznej bazie danych BigQuery, dzięki czemu możesz rzucić okiem na te dane i przeprowadzić kilka analiz bez konieczności ładowania ich wszystkich na komputer lokalny. Komputer domowy, PC lub Raspberry Pi, nie pomieści wszystkich dostępnych danych.

Rejestracja w Google dla BigQuery

Wejdź na cloud.google.com i zarejestruj się, aby skorzystać z bezpłatnego okresu próbnego. Chociaż Google wymaga karty kredytowej, aby udowodnić, że nie jesteś robotem, nie obciąży Cię nawet po zakończeniu okresu próbnego bez ręcznego przełączenia na płatne konto. Jeśli przekroczysz 300 USD podczas okresu próbnego (czego nie powinieneś), Google powiadomi Cię, ale nie obciąży Cię opłatą. Limit 300 USD na okres próbny powinien wystarczyć, abyś mógł wykonać kilka zapytań i uczyć się na platformie chmurowej BigQuery.

Czytanie Big Data Medicare

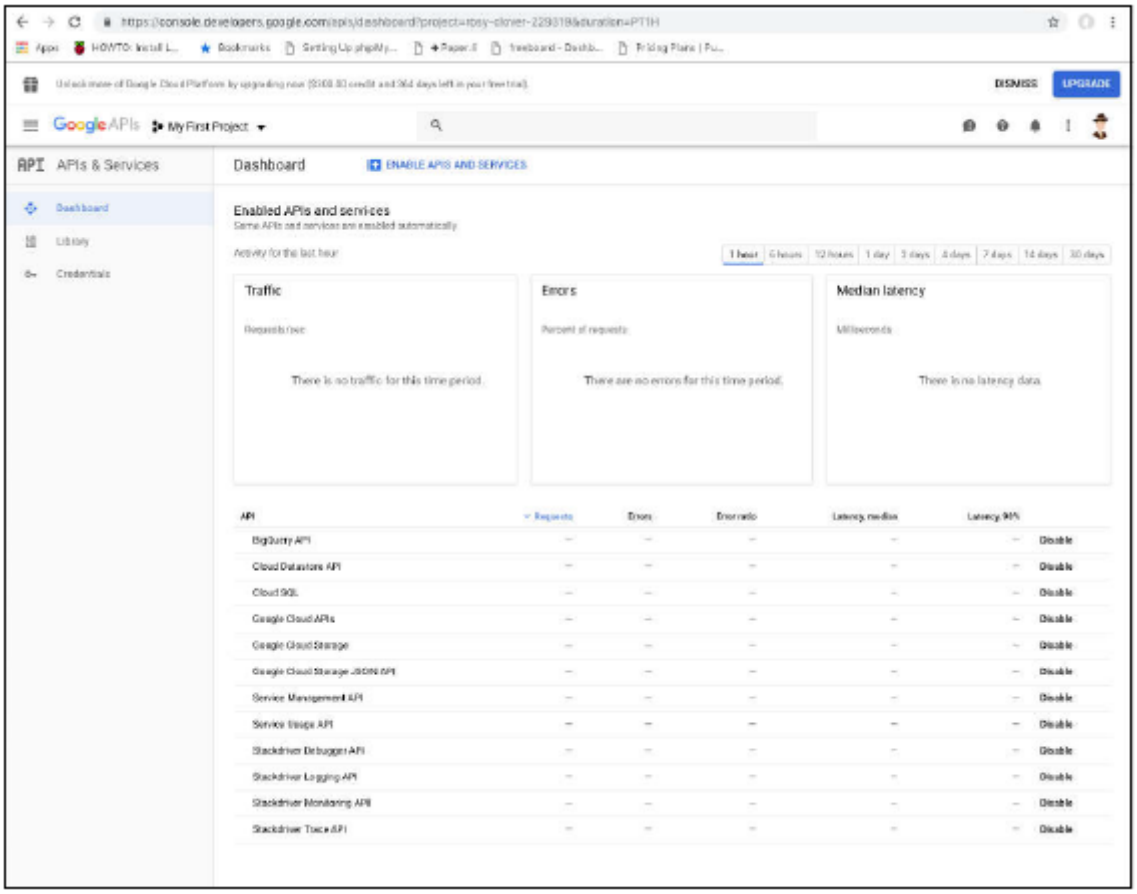
Teraz pokażemy Ci, jak skonfigurować projekt i pobrać uwierzytelniający plik .json, aby zacząć używać BigQuery we własnych programach w Pythonie.

Konfigurowanie projektu i uwierzytelnianie

Aby uzyskać dostęp do chmury Google, musisz skonfigurować projekt, a następnie otrzymać dane uwierzytelniające od Google, aby móc korzystać z ich systemów. Poniższe kroki pokażą, jak to zrobić:

1. Przejdź na stronę https://console.developers.google.com/ i zaloguj się przy użyciu nazwy konta i wygenerowanego wcześniej hasła.

2. Następnie kliknij przycisk Mój pierwszy projekt w lewym górnym rogu ekranu. Pokazuje ekran podobny do tego na rysunku



3. Kliknij przycisk Nowy projekt na wyskakującym ekranie.

4. Wpisz nazwę projektu jako MedicareProject i kliknij Utwórz.

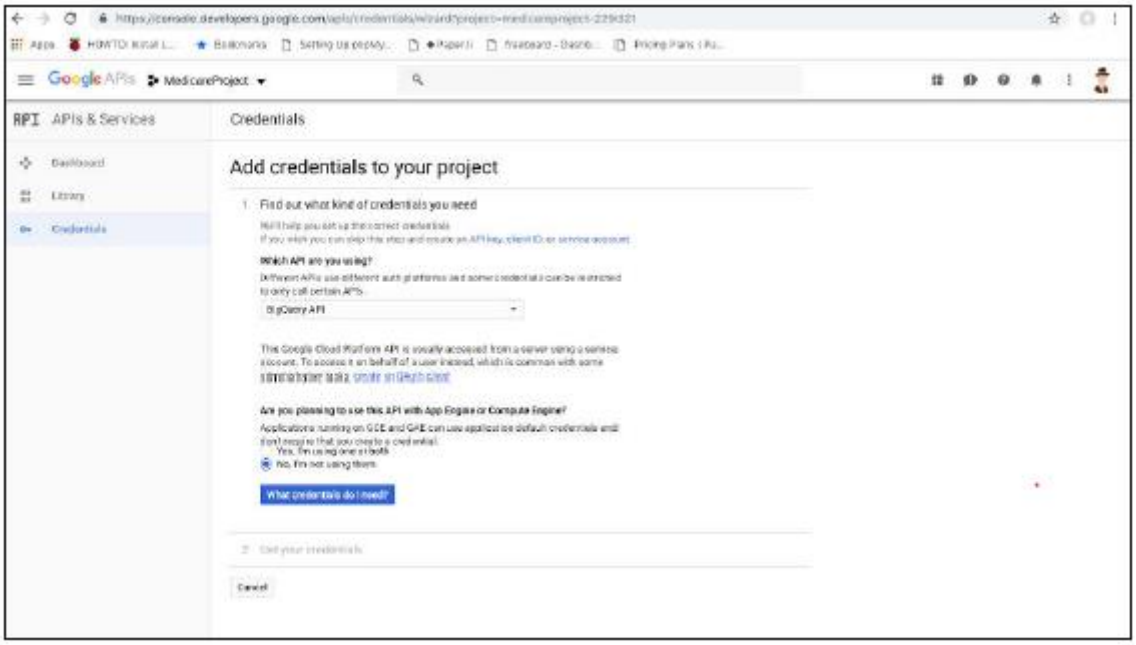
5. Następnie wybierz swój projekt, MedicareProject, za pomocą lewego górnego przycisku menu. Upewnij się, że nie zostawisz tego w domyślnym wyborze „Mój projekt". Upewnij się, że zmieniłeś go na MedicareProject - w przeciwnym razie skonfigurujesz interfejsy API i uwierzytelnianie dla niewłaściwego projektu. Jest to łatwy do popełnienia błąd.

6. Po wybraniu MedicareProject kliknij przycisk „+" u góry, aby włączyć BigQuery API.

7. Gdy pojawi się ekran wyboru API, wyszukaj BigQuery i wybierz BigQuery API. Następnie kliknij Włącz.

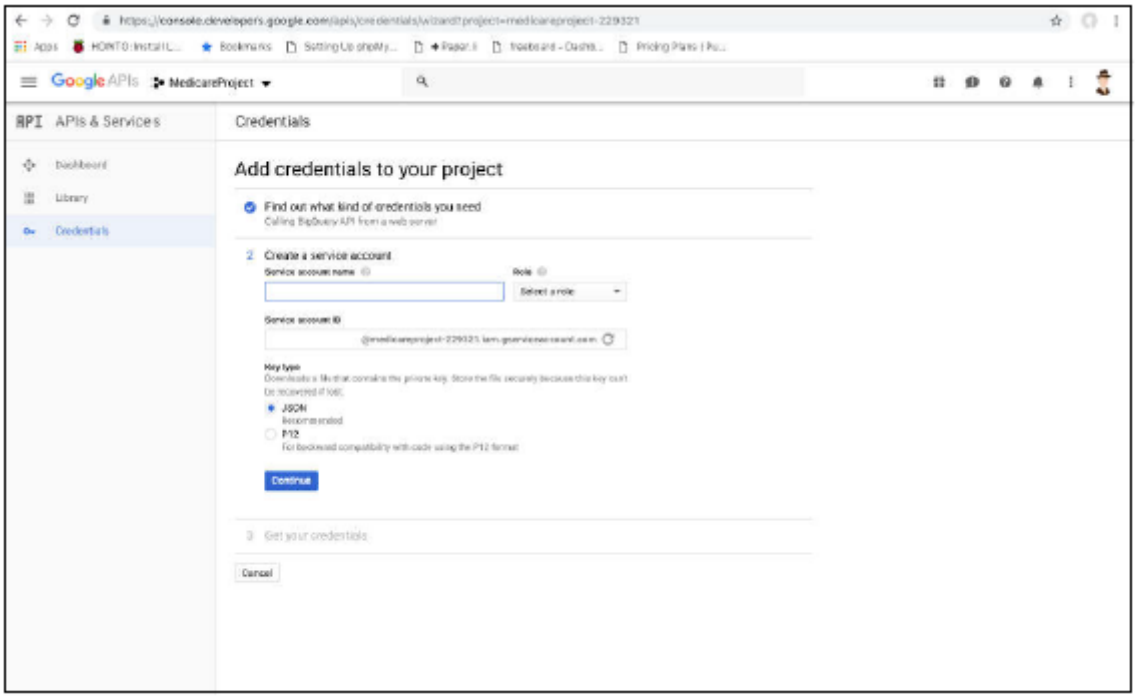
8. Teraz, aby uzyskać nasze dane uwierzytelniające. W menu po lewej stronie wybierz Poświadczenia.

Pojawi się ekran podobny do tego na rysunku



9. Wybierz BigQuery API, a następnie kliknij opcję Nie, nie używam ich w sekcji Czy planujesz używać tego interfejsu API z App Engine lub Compute Engine? Sekcja.

10. Kliknij opcję Jakich poświadczeń potrzebuję? przycisk, aby przejść do naszego ostatniego ekranu, jak pokazano na rysunku



11. Wpisz MedicareProject w polu tekstowym Service Account Name, a następnie wybierz Project Owner w menu Role.

12. Pozostaw wybrany przycisk radiowy JSON i kliknij Kontynuuj. Pojawi się komunikat informujący, że konto usługi i klucz zostały utworzone. Plik o nazwie podobnej do „MedicareProject-1223xxxxx413.json” zostanie pobrany na Twój komputer.

13. Skopiuj pobrany plik do katalogu, w którym będziesz budować plik programu w języku Python.

Pierwszy kod big data

Ten program odczytuje jeden z publicznych zestawów danych Medicare i pobiera niektóre dane do analizy. Obecnie dostępnych jest kilkadziesiąt zestawów danych, az czasem będzie ich coraz więcej. Zaczynamy od użycia zestawu danych inpatient_charges_2015. Używamy zapytania SQL, aby wybrać informacje ze zbioru danych, które chcemy przejrzeć i ostatecznie przeanalizować. (Sprawdź pobliski pasek boczny „Nauka SQL”, aby dowiedzieć się, gdzie można dowiedzieć się więcej o SQL, jeśli nie znasz jeszcze tego wszechobecnego języka zapytań).

Tabela przedstawia wszystkie kolumny w zbiorze danych inpatient_charges_2015.

Kolumna : Typ : Opis

provider_id : STRING : Numer certyfikatu CMS (CCN) dostawcy rozliczającego usługi ambulatoryjne.

nazwa_dostawcy : STRING : Nazwa dostawcy.

provider_street_address : STRING : Adres ulicy, pod którą dostawca fizycznie się znajduje.

provider_city : STRING : miasto, w którym fizycznie znajduje się dostawca

stan_dostawcy : STRING : Stan, w którym fizycznie znajduje się dostawca.

provider_zipcode : INTEGER : Kod pocztowy fizycznej lokalizacji dostawcy.

drg_definition : STRING : Kod i opis identyfikujący MS-DRG. MS-DRG to system klasyfikacji grupujący podobne stany kliniczne (diagnozy) i procedury stosowane przez szpital podczas pobytu.

hospital_referral_region_description : STRING : Region skierowania do szpitala (HRR), w którym fizycznie znajduje się usługodawca.

total_discharges : INTEGER : Liczba wypisów rozliczanych przez dostawcę za usługi szpitalne.

Average_covered_charges : FLOAT : Średnia opłata usługodawcy za usługi objęte Medicare dla wszystkich wypisów w MS-DRG. Będą się one różnić w zależności od szpitala ze względu na różnice w strukturze opłat szpitalnych.

Average_total_payments : FLOAT : Średnia suma płatności na rzecz wszystkich usługodawców w ramach MSDRG, w tym kwota MSDRG, nauczanie, nieproporcjonalny udział, kapitał i płatności odstające we wszystkich przypadkach. Średnie łączne płatności obejmują również 5 współpłatności i kwoty podlegające odliczeniu, za które odpowiada pacjent, oraz wszelkie dodatkowe płatności dokonywane przez osoby trzecie za koordynację świadczeń.

Average_medicare_payments : FLOAT : Średnia kwota, którą Medicare płaci dostawcy za udział Medicare w MS-DRG. Średnie kwoty płatności Medicare obejmują kwotę MS-DRG, nauczanie, nieproporcjonalny udział, kapitał i płatności odstające we wszystkich przypadkach. Płatności Medicare nie obejmują współpłatności beneficjenta i kwot podlegających odliczeniu ani żadnych dodatkowych płatności od stron trzecich za koordynację świadczeń.

Za pomocą nano (lub dowolnego innego edytora tekstu) wprowadź następujący kod do swojego edytora, a następnie zapisz go jako MedicareQuery1.py:

```
import pandas as pd

from google.cloud import bigquery

# set up the query

QUERY = """

SELECT provider_city, provider_state, drg_definition,

average_total_payments, average_medicare_payments

FROM `bigquery-public-data.cms_medicare.inpatient_charges_2015`

WHERE provider_city = "GREAT FALLS" AND provider_state = "MT"

ORDER BY provider_city ASC

LIMIT 1000

"""

client = bigquery.Client.from_service_account_json(

'MedicareProject2-122xxxxxf413.json')

query_job = client.query(QUERY)

df = query_job.to_dataframe()

print ("Records Returned: ", df.shape )

print ()

print ("First 3 Records")

print (df.head(3))
```

Jak tylko zbudujesz ten plik, zastąp MedicareProject2-122xxxxxf413. json z własną nazwą pliku uwierzytelniającego (którą skopiowałeś do programu katalogu wcześniej). Jeśli nie masz zainstalowanej biblioteki google.cloud, wpisz to w oknie terminala na Raspberry Pi:

```
pip3 install google-cloud-bigquery
```

Łamanie kodu

Najpierw importujemy nasze biblioteki. Zwróć uwagę na bibliotekę google.cloud i import bigquery:

```
import pandas as pd

from google.cloud import bigquery
```

Następnie konfigurujemy zapytanie SQL używane do pobierania danych, których szukamy, do Pandas DataFrame, abyśmy mogli je przeanalizować:

```
# set up the query
```

```
QUERY = """

SELECT provider_city, provider_state, drg_definition,

average_total_payments, average_medicare_payments

FROM `bigquery-public-data.cms_medicare.inpatient_charges_2015`

WHERE provider_city = "GREAT FALLS" AND provider_state = "MT"

ORDER BY provider_city ASC

LIMIT 1000

"""
```

Widzisz strukturę zapytania SQL? SELECT kolumny, które chcemy, które są podane w Tabeli 1 FROM bazy danych bigquery-public-data.cms_medicare. inpatient_charges_2015 tylko WHERE miasto_dostawcy to GREAT FALLS, a stan_dostawcy to MT. Na koniec mówimy systemowi, aby uporządkował wyniki według rosnącej kolejności alfanumerycznej według miasta_dostawcy. Co, ponieważ wybraliśmy tylko jedno miasto, jest nieco zbędne. Pamiętaj, aby zastąpić poniższą nazwę pliku json swoim plikiem uwierzytelniającym. Ten nie zadziała.

```
client = bigquery.Client.from_service_account_json(

'MedicareProject2-122xxxxxef413.json')
```

Teraz kierujemy zapytanie do chmury BigQuery:

```
query_job = client.query(QUERY)
```

I tłumaczymy wyniki naszemu dobremu przyjacielowi Pandas DataFrame:

```
df = query_job.to_dataframe()
```

Teraz tylko kilka wyników, aby zobaczyć, co otrzymaliśmy:

```
print ("Records Returned: ", df.shape )

print ()

print ("First 3 Records")

print (df.head(3))
```

Uruchom swój program za pomocą python3 MedicareQuery1.py i powinieneś zobaczyć wyniki jak poniżej. Uwaga: Jeśli pojawi się błąd uwierzytelniania, wróć i upewnij się, że umieściłeś prawidłowy plik uwierzytelniający w swoim katalogu. A jeśli to konieczne, powtórz całą procedurę generowania--uwierzytelniania, zwracając szczególną uwagę na wybór nazwy projektu.

```
Records Returned: (112, 5)

First 3 Records

provider_city provider_state

drg_ definition average_total_payments average_medicare_payments

0 GREAT FALLS MT 064 - INTRACRANIAL HEMORRHAGE OR CEREBRAL
```

INFA... 11997.11 11080.32

1 GREAT FALLS MT 039 - EXTRACRANIAL PROCEDURES W/O

CC/MCC 7082.85 5954.81

2 GREAT FALLS MT 065 - INTRACRANIAL HEMORRHAGE OR CEREBRAL

INFA... 7140.80 6145.38

Visualizing your Data

Znaleźliśmy 112 wpisów z Great Falls. Możesz wrócić i zmienić zapytanie w swoim programie, aby wybrać własne miasto i stan.

Dalej trochę analizy

Dobra, teraz nawiązaliśmy połączenie z bazą danych typu big data. Skonfigurujmy teraz inne zapytanie. Chcielibyśmy szukać pacjentów z „chorobami kości i artropatiami bez większych powikłań i chorób współistniejących". To jest kod MS_DRG 554. Odbywa się to za pomocą jednego z najbardziej tajemniczych i skomplikowanych systemów kodowania na świecie, zwanego ICD-10, który odwzorowuje praktycznie każdy stan diagnostyczny na jeden kod. Przeszukamy cały zbiór danych inpatient_charges_2015 w poszukiwaniu kodu MS_DRG 554, czyli „Choroby kości i artropatie bez poważnych komplikacji lub chorób współistniejących", czyli innymi słowy, osób, które mają problemy z kośćmi, ale obecnie nie mają poważnych problemów objawiające się na zewnątrz.

KODY ICD

ICD10 to uznana metoda kodowania profesjonalnych diagnoz medycznych do celów rozliczeniowych i analitycznych. Najnowsza wersja ICD-10 została ostatecznie wprowadzona jako obowiązkowa w 2015 roku z wielkim niepokojem w całym środowisku medycznym. Składa się w największym stopniu z ponad 155 000 kodów, od M79.603 – Ból ramienia, nieokreślony do S92.4 – Złamanie palucha większego. Kody te są nieco połączone z kodami MS_DRG, które są używane w analizowanych tutaj bazach danych Medicare, ponieważ są one używane do przyjęć do szpitali. John Shovic miał startup zajmujący się oprogramowaniem medycznym, który używał kodów ICD 10 przez dziesięć lat i miał z tymi kodami związek miłości/nienawiści. Jego ulubione kody ICD-10:

* V97.33XD: wessany do silnika odrzutowego, kolejne spotkanie.

* Z63.1: Problemy w relacjach z teściami.

* V95.43XS: Kolizja statku kosmicznego raniąca pasażera, następstwa.

* R46.1: Dziwaczny wygląd.

* Y93.D1 Aktywność, robienie na drutach i szydełku.

Kod do tego jest następujący:

```
import pandas as pd

from google.cloud import bigquery

# set up the query

QUERY = """
```

```
SELECT provider_city, provider_state, drg_definition,

average_total_payments, average_medicare_payments

FROM `bigquery-public-data.cms_medicare.inpatient_charges_2015`

WHERE drg_definition LIKE '554 %'

ORDER BY provider_city ASC

LIMIT 1000

"""

client = bigquery.Client.from_service_account_json(

'MedicareProject2-1223283ef413.json')

query_job = client.query(QUERY)

df = query_job.to_dataframe()

print ("Records Returned: ", df.shape )

print ()

print ("First 3 Records")

print (df.head(3))
```

Jedyną rzeczą, która różni się w tym programie od naszego poprzedniego, jest to, że dodaliśmy LIKE „554%”, co będzie pasować do każdej grupy DRG rozpoczynającej się od „554”.

Uruchomienie programu daje następujące wyniki:

```
Records Returned:  (286, 5)

First 3 Records
  provider_city provider_state                              drg_definition
   average_total_payments  average_medicare_payments
0      ABINGTON             PA  554 - BONE DISEASES & ARTHROPATHIES W/O MCC
   5443.67                    3992.93
1        AKRON              OH  554 - BONE DISEASES & ARTHROPATHIES W/O MCC
   5581.00                    4292.47
2       ALBANY              NY  554 - BONE DISEASES & ARTHROPATHIES W/O MCC
   7628.04                    5137.31
```

Teraz mamy kilka interesujących danych. Zróbmy małą analizę. Jaki procent całkowitych opłat za to schorzenie pokrywa Medicare (pozostałą część pokrywa pacjent)? Kod do tego będzie (nazwijmy go MedicareQuery3.py):

```
import pandas as pd

from google.cloud import bigquery

# set up the query

QUERY = """

SELECT provider_city, provider_state, drg_definition,
```

```
average_total_payments, average_medicare_payments

FROM `bigquery-public-data.cms_medicare.inpatient_charges_2015`

WHERE drg_definition LIKE '554 %'

ORDER BY provider_city ASC

LIMIT 1000

"""

client = bigquery.Client.from_service_account_json(

'MedicareProject2-1223283ef413.json')

query_job = client.query(QUERY)

df = query_job.to_dataframe()

print ("Records Returned: ", df.shape )

print ()

total_payment = df.average_total_payments.sum()

medicare_payment = df.average_medicare_payments.sum()

percent_paid = ((medicare_payment/total_payment))*100

print ("Medicare pays {:4.2f}% of Total for 554 DRG".format(percent_paid))

print ("Patient pays {:4.2f}% of Total for 554 DRG".format(100-percent_paid))
```

A wyniki:

```
Records Returned: (286, 5)

Medicare pays 77.06% of Total for 554 DRG

Patient pays 22.94% of Total for 554 DRG
```

Procent płatności według stanu

Teraz w tym programie wybieramy unikalne stany w naszej bazie danych (nie wszystkie stany są reprezentowane) i iterujemy po stanach, aby obliczyć procent płacony przez Medicare według stanu za 554. Nazwijmy ten MedicareQuery4.py:

```
import pandas as pd

from google.cloud import bigquery

# set up the query

QUERY = """

SELECT provider_city, provider_state, drg_definition,

average_total_payments, average_medicare_payments

FROM `bigquery-public-data.cms_medicare.inpatient_charges_2015`
```

```
WHERE drg_definition LIKE '554 %'
ORDER BY provider_city ASC
LIMIT 1000
"""
client = bigquery.Client.from_service_account_json(
'MedicareProject2-1223283ef413.json')
query_job = client.query(QUERY)
df = query_job.to_dataframe()
print ("Records Returned: ", df.shape )
print ()
# find the unique values of State
states = df.provider_state.unique()
states.sort()
total_payment = df.average_total_payments.sum()
medicare_payment = df.average_medicare_payments.sum()
percent_paid = ((medicare_payment/total_payment))*100
print("Overall:")
print ("Medicare pays {:4.2f}% of Total for 554 DRG".format(percent_paid))
print ("Patient pays {:4.2f}% of Total for 554 DRG".format(100-percent_paid))
print ("Per State:")
# now iterate over states
print(df.head(5))
state_percent = []
for current_state in states:
state_df = df[df.provider_state == current_state]
state_total_payment = state_df.average_total_payments.sum()
state_medicare_payment = state_df.average_medicare_payments.sum()
state_percent_paid = ((state_medicare_payment/state_total_payment))*100
state_percent.append(state_percent_paid)
print ("{:s} Medicare pays {:4.2f}% of Total for 554 DRG".format
(current_state,state_percent_paid))
```

A teraz trochę wizualizacji

W naszym ostatnim eksperymencie zwizualizujmy dane stan po stanie na wykresie za pomocą MatPlotLib. Przechodząc do naszego programu VNC, aby mieć GUI na naszym Raspberry Pi, dodajemy następujący kod na końcu poprzedniego kodu Medicare Query4.py:

```
# we could graph this using MatPlotLib with the two lists

# but we want to use DataFrames for this example

data_array = {'State': states, 'Percent': state_percent}

df_states = pd.DataFrame.from_dict(data_array)

# Now back in dataframe land

import matplotlib.pyplot as plt

import seaborn as sb

print (df_states)

df_states.plot(kind='bar', x='State', y= 'Percent')

plt.show()
```

Czy masz już seaborn na swoim Raspberry Pi (a jeśli zainstalowałeś MatPlotLib, prawdopodobnie już to robisz)? Aby się dowiedzieć, uruchom MedicareQuery4. py przykładowy program w Pythonie. Jeśli nie masz zainstalowanego seaborne, uruchom następujące polecenie:

```
sudo apt-get install python3-seaborn
```

Wyszukiwanie najbardziej zanieczyszczonego miasta na świecie w ujęciu godzinowym

Jeszcze jeden szybki przykład. Istnieje inna publiczna baza danych na BigQuery, OpenAQ, która zawiera pomiary jakości powietrza z 47 krajów na całym świecie. A ta baza danych jest aktualizowana co godzinę, wierzcie lub nie.

Oto kod, który wybiera trzy najbardziej zanieczyszczone miasta na świecie pod względem jakości powietrza:

```
import pandas as pd

from google.cloud import bigquery

# sample query from:

QUERY = """

SELECT location, city, country, value, timestamp

FROM `bigquery-public-data.openaq.global_air_quality`

WHERE pollutant = "pm10" AND timestamp > "2017-04-01"

ORDER BY value DESC

LIMIT 1000

"""
```

```
client = bigquery.Client.from_service_account_json(

'MedicareProject2-1223283ef413.json')

query_job = client.query(QUERY)

df = query_job.to_dataframe()

print (df.head(3))
```

Skopiuj ten kod do pliku o nazwie PollutedCity.py i uruchom program. Obecny wynik uruchomienia kodu (w chwili pisania) był następujący:

```
        location          city country  value    timestamp
0       Dilovası        Kocaeli   TR  5243.00 2018-01-25 12:00:00+00:00
1  Bukhiin urguu    Ulaanbaatar   MN  1428.00 2019-01-21 17:00:00+00:00
2  Chaiten Norte  Chaiten Norte   CL   999.83 2018-04-24 11:00:00+00:00
```

Wygląda na to, że Dilovasi, Kocaeli, Turcja nie jest teraz zdrowym miejscem. Szybkie wyszukiwanie w Google Dilovasi stwierdza, że wskaźniki raka są trzy razy wyższe niż średnia światowa. Ta uderzająca różnica najwyraźniej wynika z zanieczyszczenia środowiska metalami ciężkimi utrzymującego się na tym obszarze od około 40 lat, głównie z powodu intensywnej industrializacji. Na pewno będę to sprawdzać codziennie.