دوره دیتا ساینس کاربردی

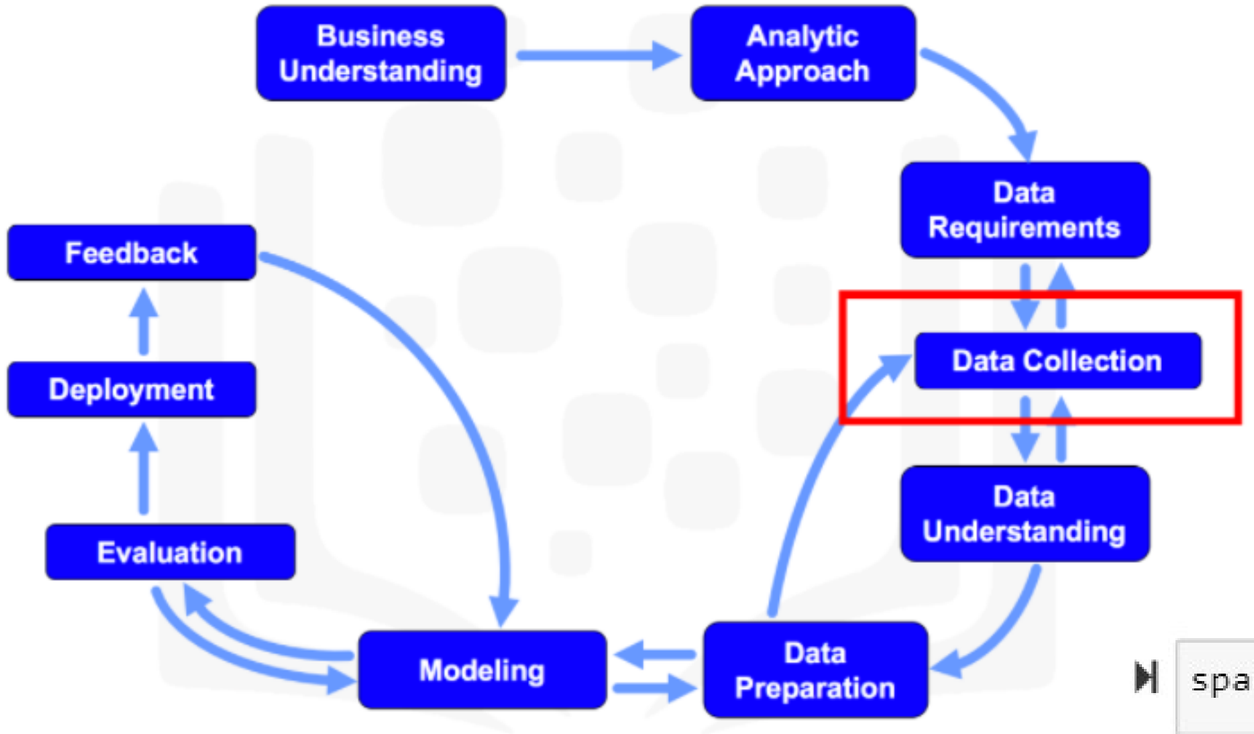
Web Scraping & SQL

dataroadmap

مدرس: مونا حاتمی

جلسه دوازدهم

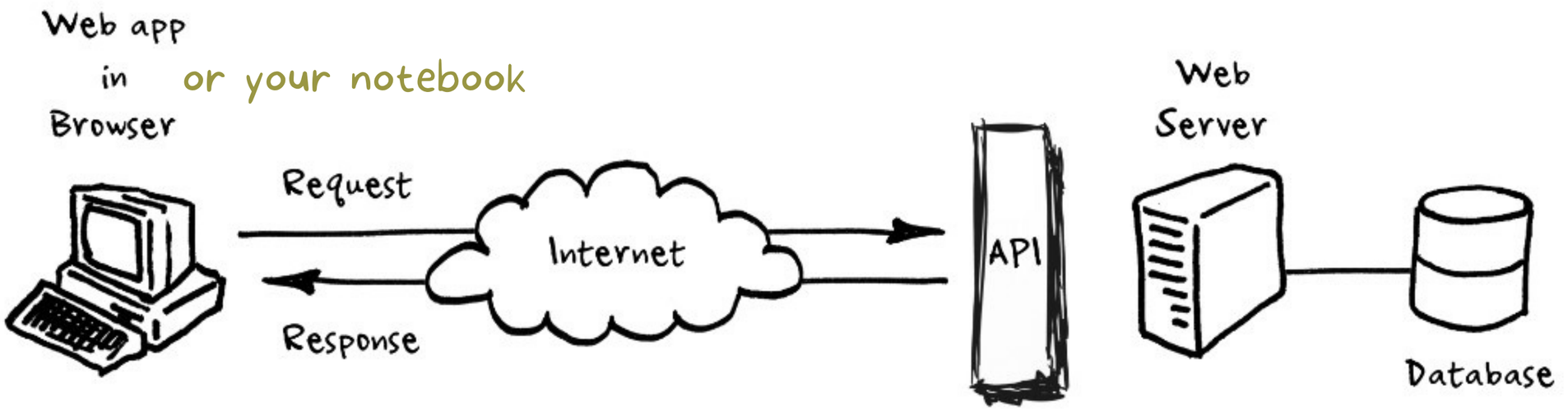
Capston Project Overview




```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

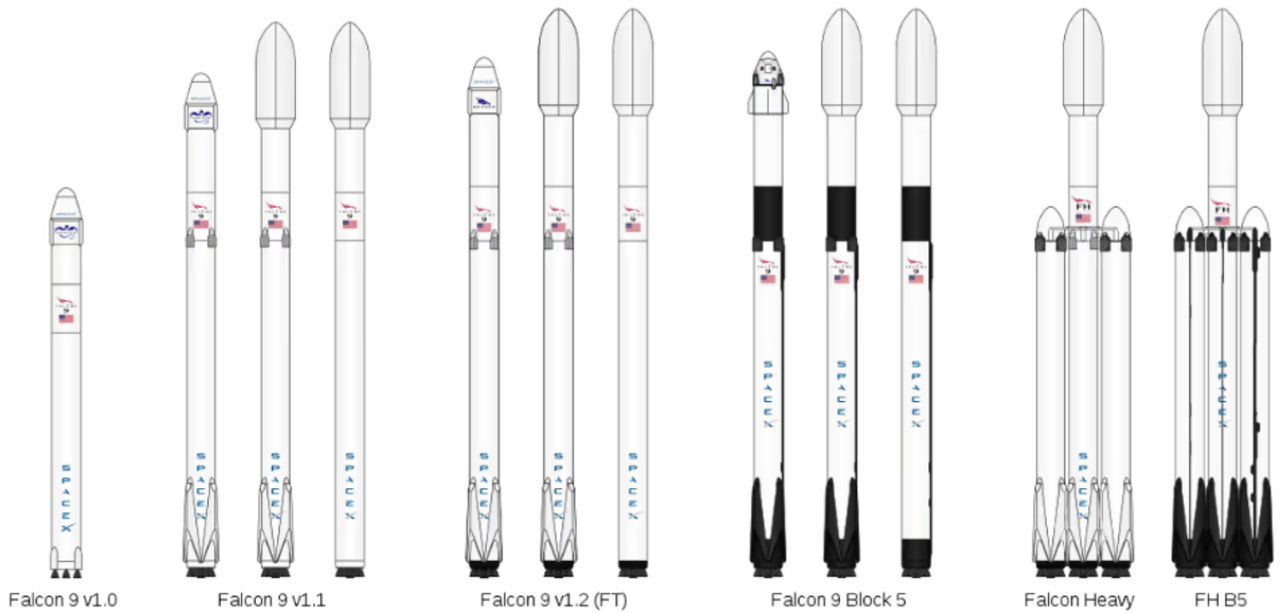
```
response = requests.get(spacex_url)
```

Data Collection from API



Data Collection from Webscraping

Data Collection from Wikipedia



List of Falcon 9 and Falcon Heavy launches

Since June 2010, rockets from the Falcon 9 family have been launched 191 times, with 189 full mission successes, one partial failure and one total loss of the spacecraft. In addition, one rocket and its payload were destroyed on the launch pad during the fueling process before a static fire test was set to occur.

W Wikipedia / Dec 5

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Data Collection View Page Source

2010 to 2019 [edit]

For launches prior to 2020, please refer to List of Falcon 9 and Falcon Heavy launches (2010–2019).

2020 [edit]

In late 2019, Gwynne Shotwell stated that SpaceX hoped for as many as 24 launches for Starlink satellites in 2020,[10] in addition to 14 or 15 non-Starlink launches. At 26 launches, 14 of which were for Starlink satellites, Falcon 9 had its most pr
rockets were second most prolific rocket family of 2020, only behind China's Long March rocket family.[11]

[hide] Flight No.	Date and time (UTC)	Version, booster[b]	Launch site	Payload[c]			Customer	Launch outcom
78	7 January 2020, 02:19:21[12]	F9 B5 △ B1049.4	CCAFS, SLC-40	Starlink 2 v1.0 (60 satellites)			paceX	Succes
	Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included						with ground-based astronomical observations.[1	
79	19 January 2020, 15:30[14]	F9 B5 △ B1046.4	KSC, LC-39A	Crew Dragon in-flight abort test[15] (Dragon C205.1)			ASA (CTS)[17]	Succes
	An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule;[18] but th crewed flight.[20] As expected, the booster was destroyed by aerodynamic forces after the capsule aborte						lashed down in the ocean 31 km (19 mi) downr 019.[19] The abort test used the capsule original had a mass simulator in place of its engine.	
80	29 January 2020, 14:07[22]	F9 B5 △ B1051.3	CCAFS, SLC-40	Starlink 3 v1.0 (60 satellites)			paceX	Succes
	Third operational and fourth large batch of Starlink satellites, deployed in a circular 290 km (180 mi) orbit						n.[23]	
81	17 February 2020, 15:05[24]	F9 B5 △ B1056.4	CCAFS, SLC-40	Starlink 4 v1.0 (60 satellites)			paceX	Succes
	Fourth operational and fifth large batch of Starlink satellites. Used a new flight profile which deployed in booster failed to land on the drone ship[25] due to incorrect wind data.[26] This was the first time a flight p						circular orbit and firing the second stage engine	
82	7 March 2020, 04:50[27]	F9 B5 △ B1059.2	CCAFS, SLC-40	SpaceX CRS-20 (Dragon C112.3 △)	1,977 kg (4,359 lb)[28] (excl. Dragon mass)	LEO (ISS)	NASA (CRS)	Succes
	Last launch of phase 1 of the CRS contract. Carries *Bartolomeo*, an ESA platform for hosting external payloads onto ISS.[29] Originally scheduled to launch on 2 March 2020, the launch date was pushed back due to a second stage er decided to swap out the second stage instead of replacing the faulty part.[30] It was SpaceX's third flight of the Dragon C112 and the last launch of the cargo Dragon spacecraft							

Data Collection Page Source

```
</td></tr></tbody></table></div>
<div>
<table class="multicol" role="presentation" style="border-collapse: collapse; padding: 0
<tr>
<td style="text-align: left; vertical-align: top;">
<h3><span class="mw-headline" id="Launch_outcomes">Launch outcomes</span><span class="mw
<div style="margin-top:1em;max-width:480px;" class="chart noresize">
<div style="position:relative;min-height:320px;min-width:480px;max-width:480px;">
<div style="float:right;position:relative;min-height:240px;min-width:380px;max-width:380
<div style="position:absolute;left:165px;top:237px;height:2px;min-width:19px;max-width:1
<div style="position:absolute;left:138px;top:237px;height:2px;min-width:19px;max-width:1
<div style="position:absolute;left:57px;top:237px;height:2px;min-width:19px;max-width:19
<div style="position:absolute;left:3px;top:233px;height:6px;min-width:19px;max-width:19p
<div style="position:absolute;left:57px;top:234px;height:2px;min-width:19px;max-width:19
<div style="position:absolute;left:84px;top:230px;height:9px;min-width:19px;max-width:19
<div style="position:absolute;left:111px;top:219px;height:20px;min-width:19px;max-width:
```

Import Libraries

```
!pip install beautifulsoup4
!pip install requests
!pip install unicodedata
```

```
import requests
from bs4 import BeautifulSoup
import unicodedata
import pandas as pd
```

Request the Falcon9 Launch Wiki page from its url

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

```
page=requests.get(static_url)
```

```
page.status_code
```

```
4]: 200
```

```
page.text
```

```
]: '<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta charset="UTF-8"/>\n
con 9 and Falcon Heavy launches - Wikipedia</title>\n<script>document.documentElement.className="cl
{"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefau
y","wgMonthNames":["","January","February","March","April","May","June","July","August","September"
er","December"],"wgRequestId":"49b0886a-92a4-4f60-baa5-f77d04114efb","wgCSPNonce":false,"wgCanonica
```

Create a BeautifulSoup object from the HTML response

```
soup = BeautifulSoup(page.text, 'html.parser')
```

```
soup
```

```
<!DOCTYPE html>

<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparat
["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January
h","April","May","June","July","August","September","October","November","December"],"wgRequest
f-bcd4-8deb3f924cea","wgCSPNonce":false,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":
er":0,"wgPageName":"List_of_Falcon_9_and_Falcon_Heavy_launches","wgTitle":"List of Falcon 9 and
s","wgCurRevisionId":1125345595,"wgRevisionId":1027686922,"wgArticleId":37574004,"wgIsArticle":
lse,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Source attributio
dead external links","Articles with dead external links from February 2021","Articles with perm
```

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Extract Target Table

```
html_tables=soup.find_all('table')
```

tarting from the third table is our target table contains the actual launch records.

```
html_tables[2]
```

```
: <table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date andtime ()
</th>
<th scope="col">
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
```

Extract Columns Name

```
column_names = []
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for i in first_launch_table.find_all('th'):
    if extract_column_from_header(i)!=None:
        if len(extract_column_from_header(i))>0:
            column_names.append(extract_column_from_header(i))
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Create a data frame by parsing the launch HTML tables

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
```

Functions

```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```

Create a dataframe by parsing the launch HTML tables

In [13]:

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            print(time)
```

Convert Dictionary to Dataframe

```
df=pd.DataFrame(launch_dict)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Flight No.       121 non-null    object
 1   Launch site      121 non-null    object
 2   Payload          121 non-null    object
 3   Payload mass     121 non-null    object
 4   Orbit            121 non-null    object
 5   Customer         120 non-null    object
 6   Launch outcome   121 non-null    object
```

```
df
```

3]:

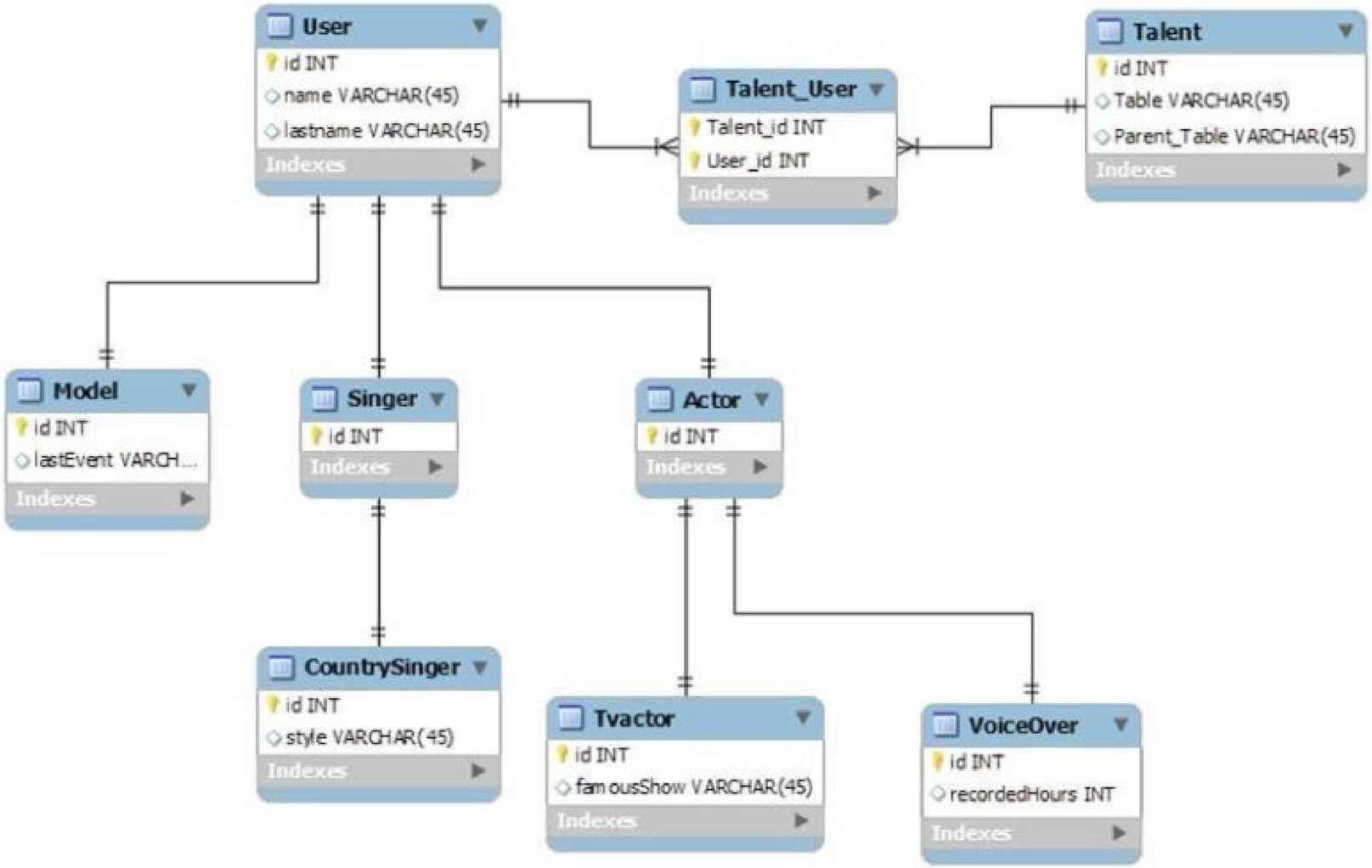
	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

121 rows × 11 columns

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

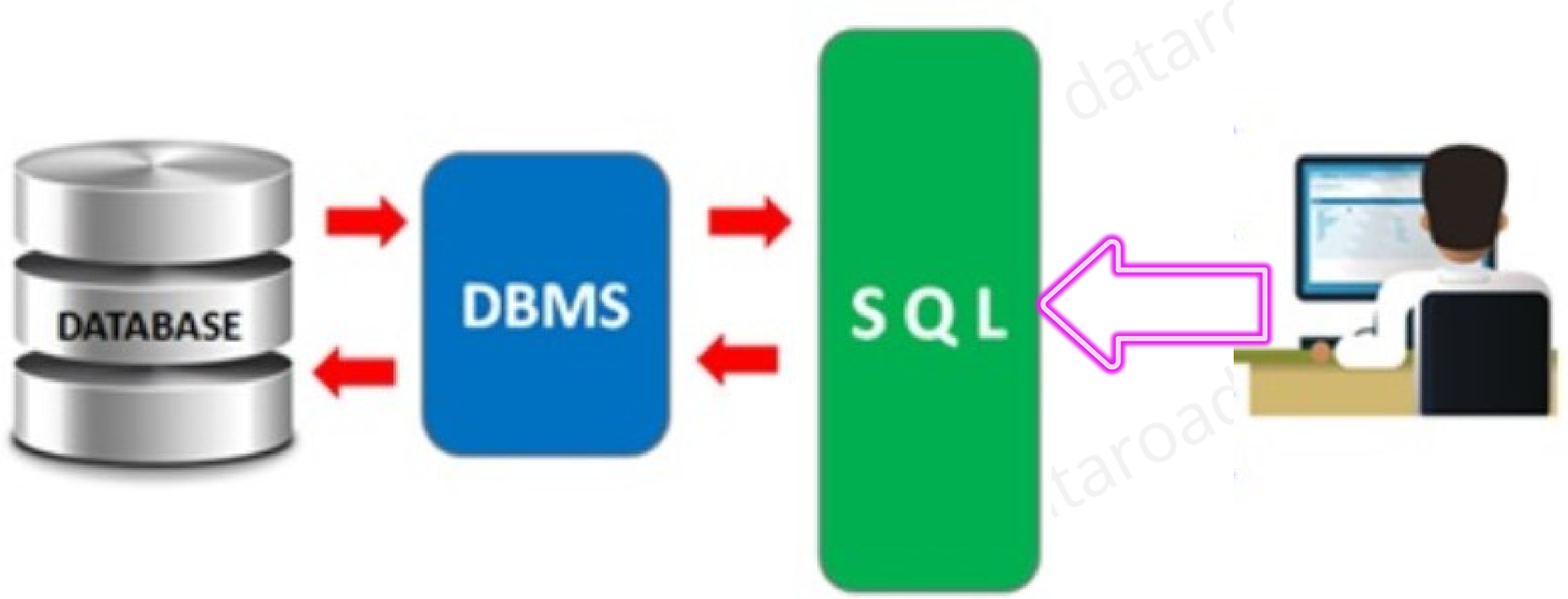
Introuduction to Database

Introuduction to Database

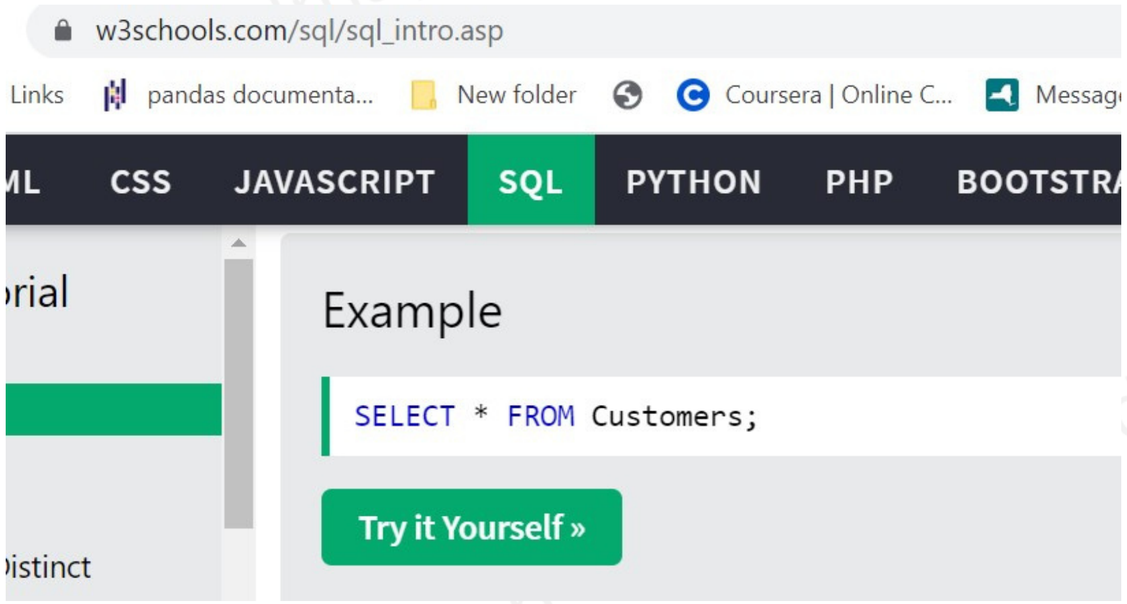


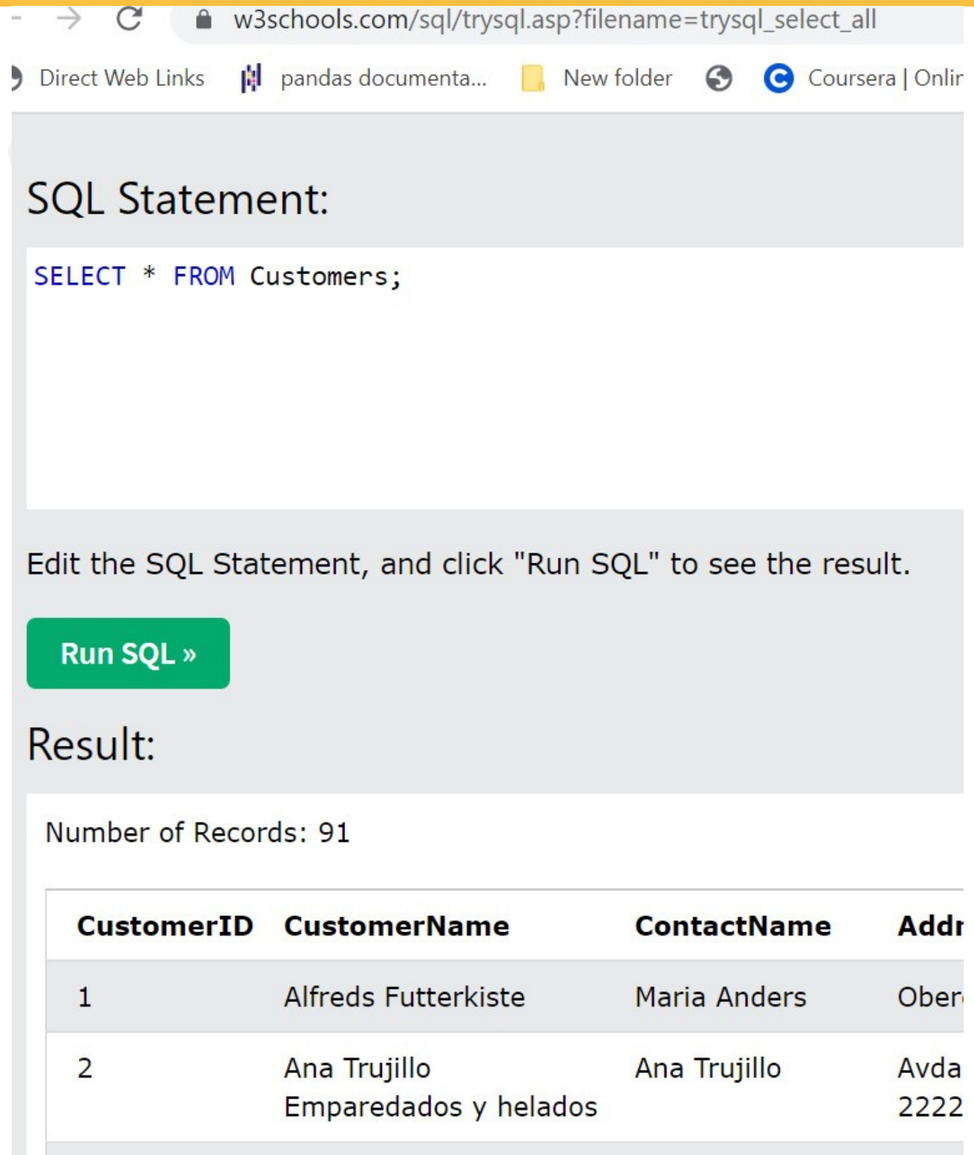
Database Managment Systems

Database Managment Systems

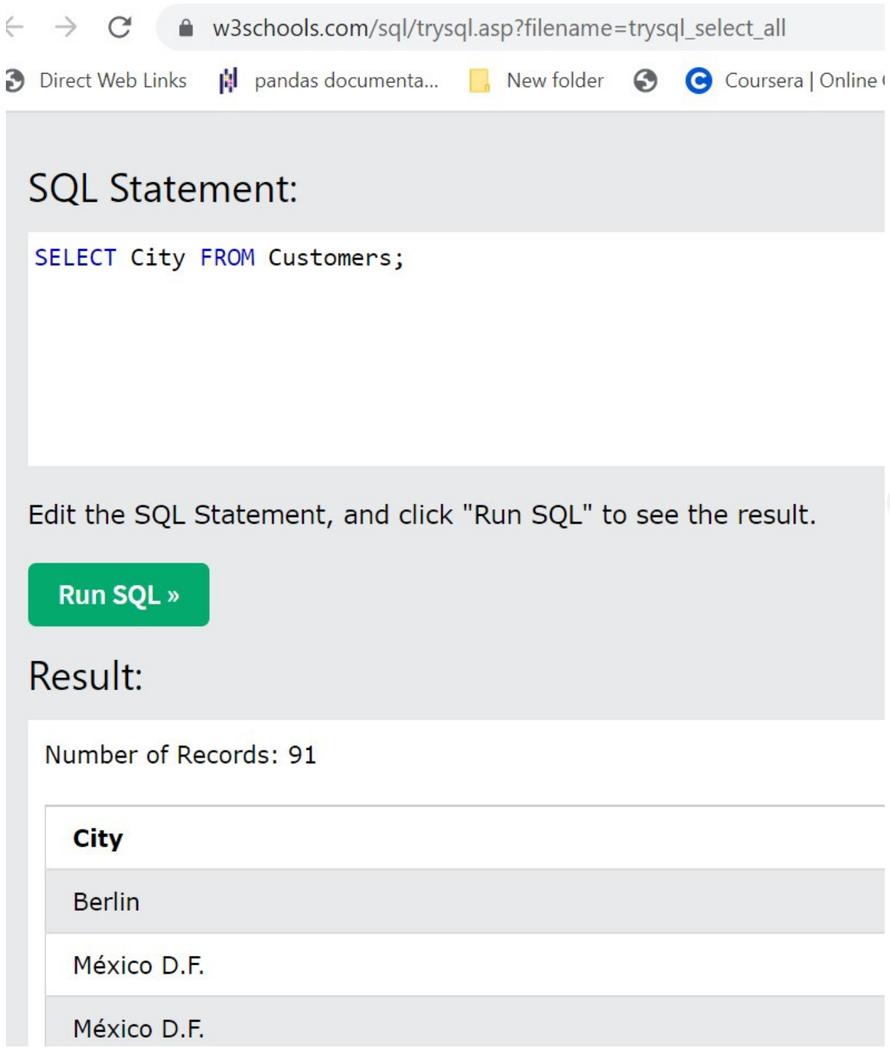


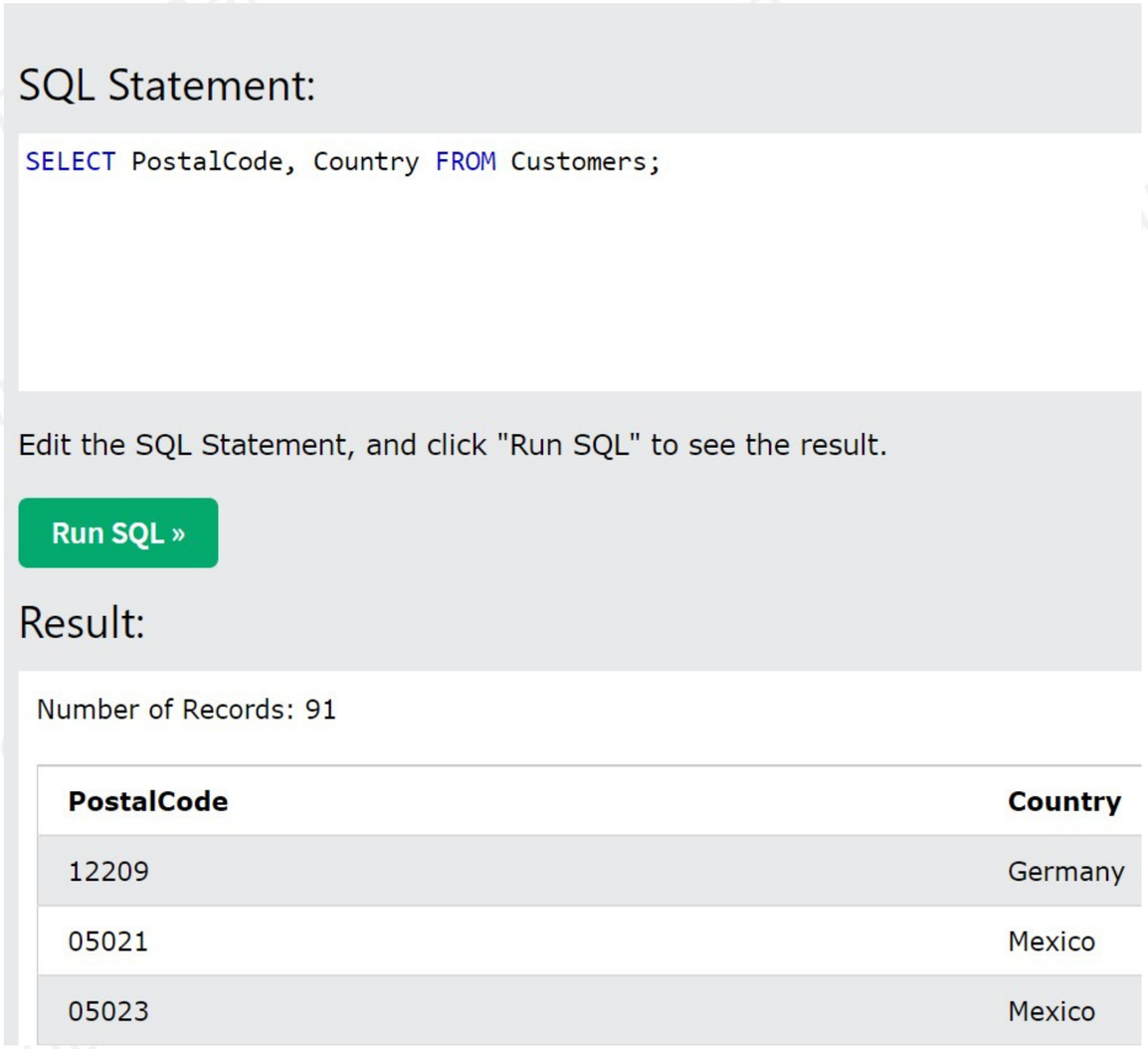
SQL(Structured Query Language)





SQL


w3schools.com/sql/trysql.asp?filename=trysql_select_all
Direct Web Links
pandas documenta...
New folder
Coursera | Online
SQL Statement:
SELECT City FROM Customers;
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:
Number of Records: 91
City
Berlin
México D.F.
México D.F.


SQL Statement:
SELECT PostalCode, Country FROM Customers;
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:
Number of Records: 91
PostalCode
Country
12209
Germany
05021
Mexico
05023
Mexico

SQL

SQL Statement:

```
SELECT PostalCode, Country FROM Customers
WHERE PostalCode<10000;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 19

PostalCode	Country
05021	Mexico
05023	Mexico
05022	Mexico

SQL-Where

SQL Statement:

```
SELECT PostalCode, Country FROM Customers
Where PostalCode<10000 and Country=='Mexico' ;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

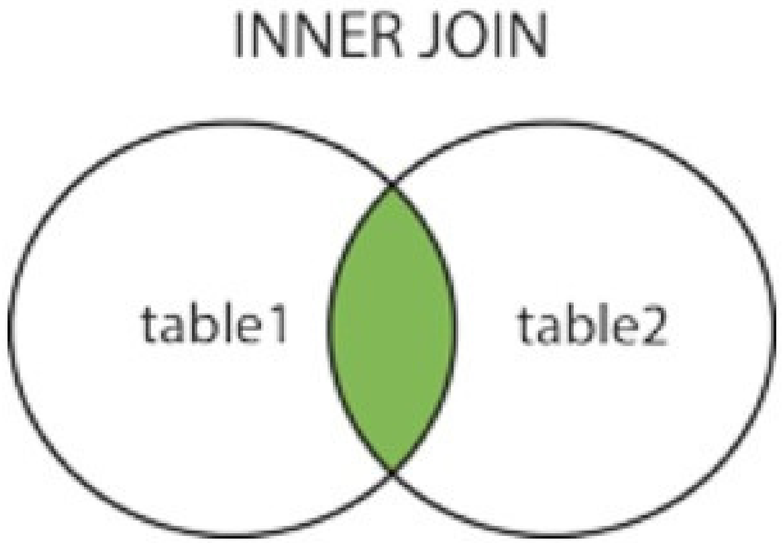
Run SQL »

Result:

Number of Records: 5

PostalCode	Country
05021	Mexico
05023	Mexico
05022	Mexico

SQL-INNER JOIN



SQL Statement:

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

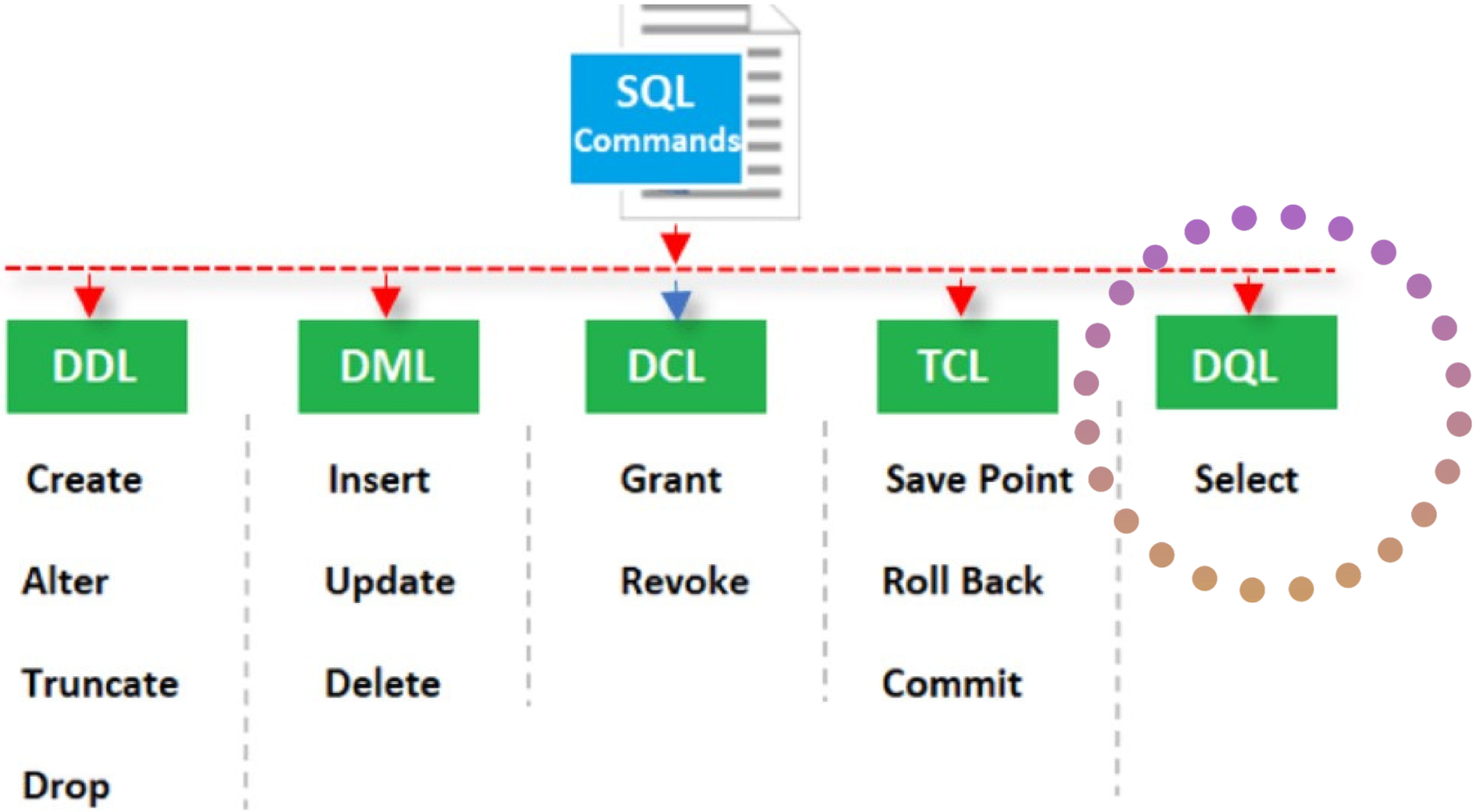
Run SQL »

Result:

Number of Records: 196

OrderID	CustomerName
10248	Wilman Kala
10249	Tradição Hipermercados
10250	Hanari Carnes

SQL Commands



SQL

Cheat Sheet

SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not

Assignment:

تمرین:

کدهای ارائه شده در درس را بررسی و اجرا کنید.

در وبسایت https://www.w3schools.com, سی کو ال و پایتون تمرین کنید.