CSCI 621 : DATABASE SYSTEM IMPLEMENTATION
NoSQL TERM PAPER - SIMPLEDB
IMPLEMENTATION AND WORKLOAD DISTRIBUTION

Group 6:
Amanraj Lnu
Kruthi Nagabhushan
Aarti Nayak
Swetna Tribhuvan

About the application - Used_Cars app:

The application provides users to trade cars on an online platform. One can search for the car of his choice and add his car as a new listing. The front-end of the program is developed with HTML and CSS, the database is SimpleDB, and the back-end is flask. As we know SimpleDB is a product or the service provided by AWS, the application needs a specific python interface to run called boto.

I. Front-end :

The front end consists of a few HTML web pages designed for different functions. The application has

- Home.html : It is the home page of the application. It is linked to style.css for styling. It has 2 navigation buttons -
  - ➔ View cars : Navigates to the ViewCar.html page
  - ➔ Sell my car : Navigates to sell my car section of the home page. This section consists of a single form with all the fields required in order to have a successful listing of the car. When the listing is successful, the application generates a unique 9 digit listing ID for the user with which he can retrieve his listing from the database. In order to have the successful listing, the user has to provide valid inputs such as VIN of length 17, Dealer Zip of 5 digits etc.

Figure:1-home.html

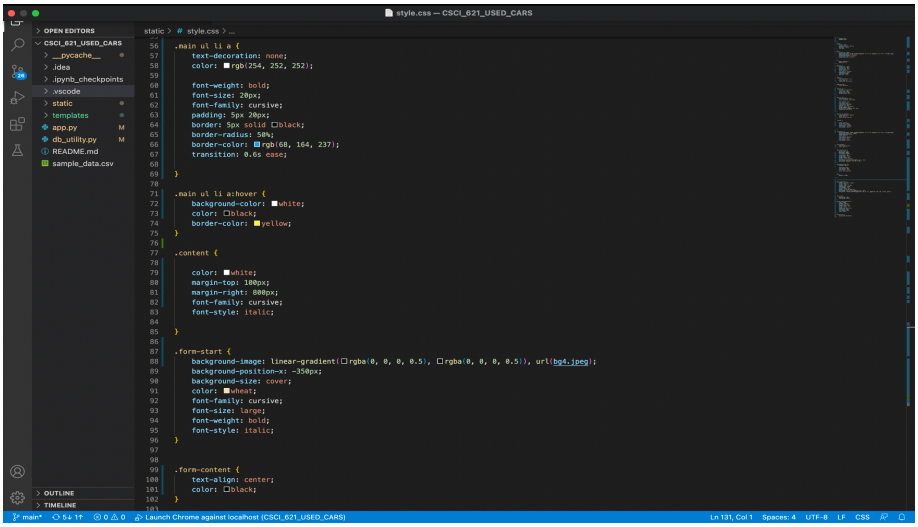


Figure2: style.css

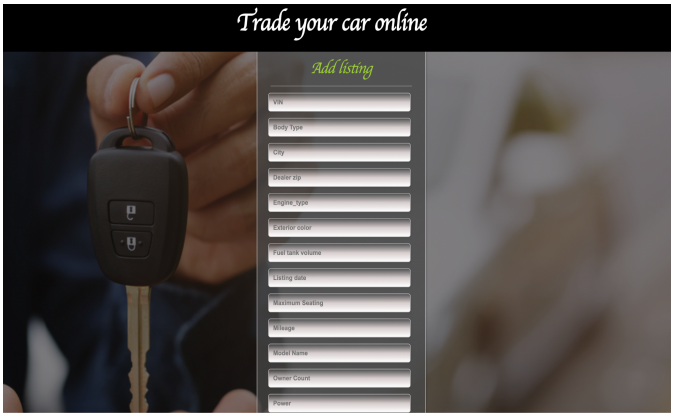


Figure 3: Sell my car section of home.html

- viewCar.html: This the second webpage which is styled using style_viewCar.css. This page helps the user to look up the car according to the listing ID or model name of the car or according to his preferences. There are 4 forms for these respectively.

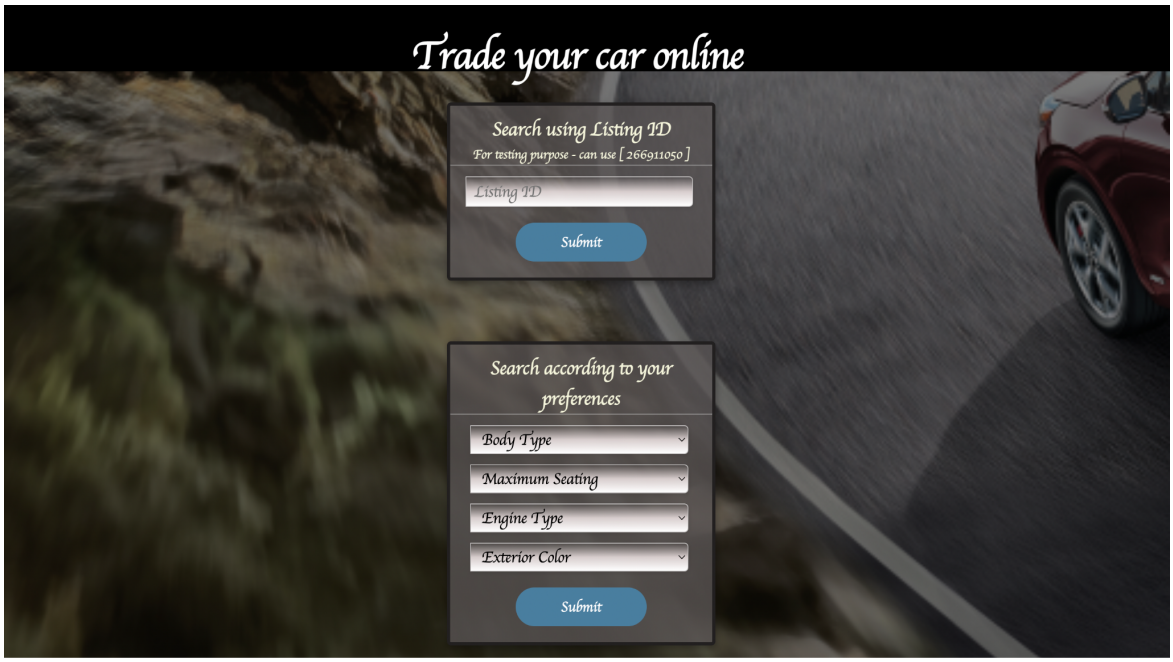


Figure 4

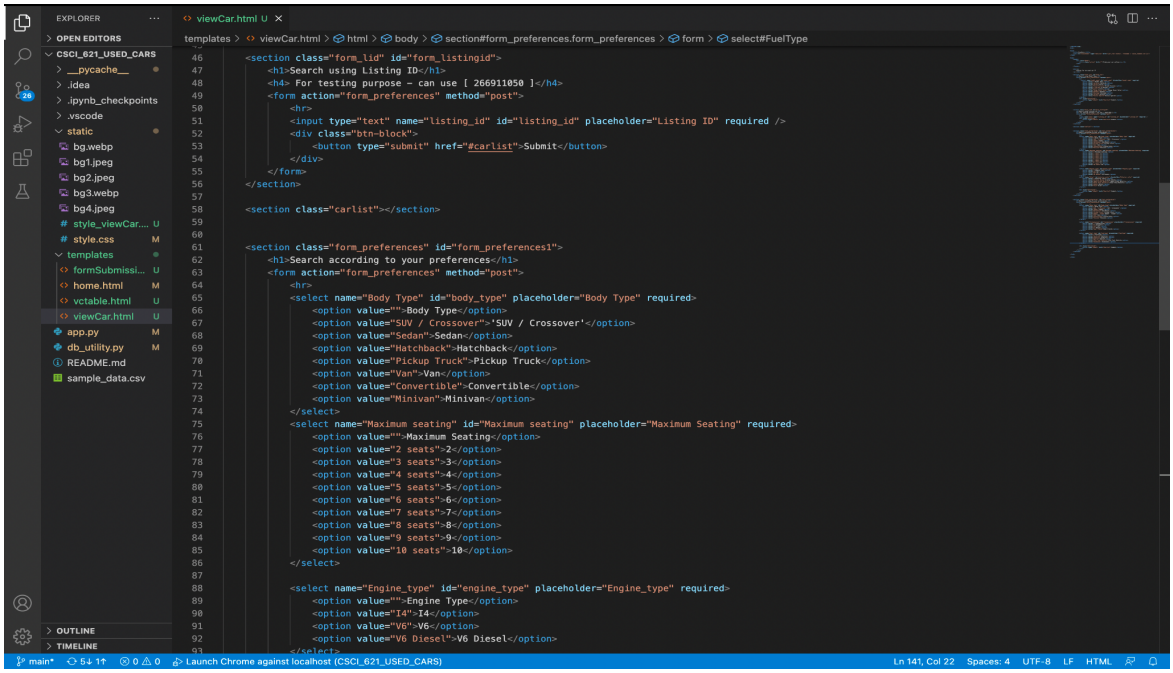
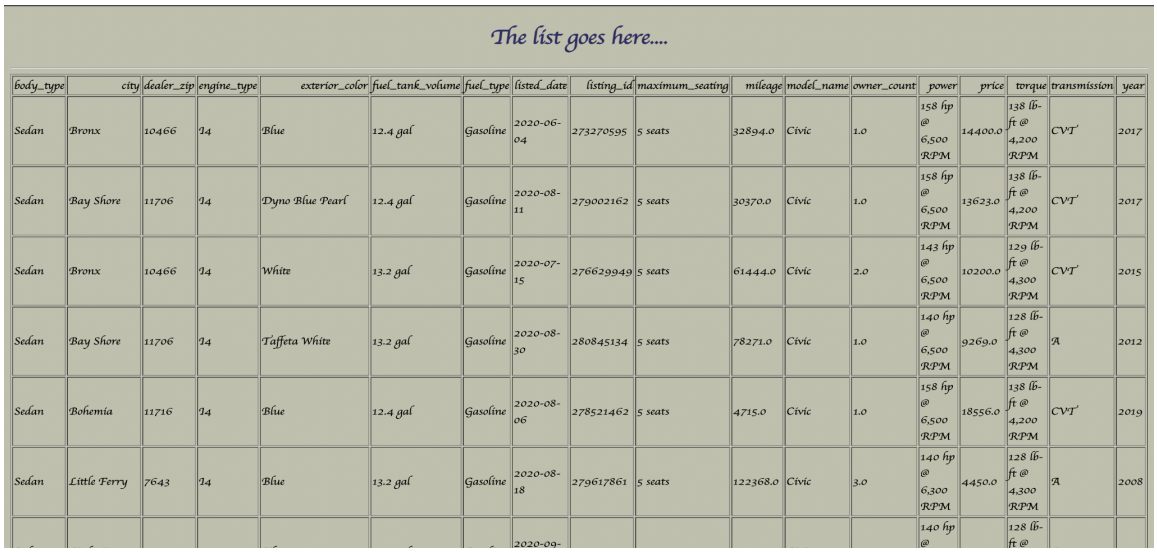


Figure 5: viewCar.html

- formSubmissionStatus.html: This page turns up when the user adds a new listing with all the valid inputs as mentioned above. This page displays message along with the new unique 9 digit listing ID.

Figure 6: formSubmissionStatus.html

- Vctable.html: This page displays all the listings which match the filters applied/required by the user in one of the forms on viewCars.html page in table format.



body_type	city	dealer_zip	engine_type	exterior_color	fuel_tank_volume	fuel_type	listed_date	listing_id	maximum_seating	mileage	model_name	owner_count	power	price	torque	transmission	year
Sedan	Bronx	10466	I4	Blue	12.4 gal	Gasoline	2020-06-04	273270595	5 seats	32894.0	Civic	1.0	158 hp @ 6,500 RPM	14400.0	138 lb-ft @ 4,200 RPM	CVT	2017
Sedan	Bay Shore	11706	I4	Dyno Blue Pearl	12.4 gal	Gasoline	2020-08-11	279002162	5 seats	30370.0	Civic	1.0	158 hp @ 6,500 RPM	13623.0	138 lb-ft @ 4,200 RPM	CVT	2017
Sedan	Bronx	10466	I4	White	13.2 gal	Gasoline	2020-07-15	276629949	5 seats	61444.0	Civic	2.0	143 hp @ 6,500 RPM	10200.0	129 lb-ft @ 4,300 RPM	CVT	2015
Sedan	Bay Shore	11706	I4	Taffeta White	13.2 gal	Gasoline	2020-08-30	280845134	5 seats	78271.0	Civic	1.0	140 hp @ 6,500 RPM	9269.0	128 lb-ft @ 4,300 RPM	A	2012
Sedan	Bohemia	11716	I4	Blue	12.4 gal	Gasoline	2020-08-06	278521462	5 seats	4715.0	Civic	1.0	158 hp @ 6,500 RPM	18556.0	138 lb-ft @ 4,200 RPM	CVT	2019
Sedan	Little Ferry	7643	I4	Blue	13.2 gal	Gasoline	2020-08-18	279617861	5 seats	122368.0	Civic	3.0	140 hp @ 6,300 RPM	4450.0	128 lb-ft @ 4,300 RPM	A	2008
							2020-09-						140 hp @		128 lb-ft @		

Figure 7

II. Flask :

It is the web framework that's being used in this application to render all the above explained HTML web pages in a web browser on localhost port 5000. It's being implemented on app.py python file.

- app.py: In this file, we have various functions for each different purpose. These functions can be bound to certain URL using route() provided by Flask. Also route( ) is used handle HTTP method i.e post method to access form data in this application. We also see each function rendering HTML templates using render_template method. Keeping app.py as a simple file, we have a handler file called db_utilities.py.

- db_utilities.py : In this file, we have implemented the available boto.sdb interface to connect with AWS's SimpleDB.

```
import boto.sdb
```

Figure 8

Also, in this file we have functions to

➔ validate the inputs which is necessary to have a successful listing

```
def validate_form_submitted_data(form_data):
    for key in form_data:
        if key in VALIDATION_FUNCTION_MAPPING:
            validation_function = VALIDATION_FUNCTION_MAPPING.get(key)
            if callable(validation_function):
                check, message = validation_function(form_data[key])
                if not check:
                    return FORM_FAILURE_MESSAGE.format(message)

    query = construct_insert_query(form_data)
    put_data_into_db(None,query)
    return FORM_SUCCESS_MESSAGE.format(query[form_data['vin']].get('listing_id'))
```

Figure 9

➔ generate randon unique listing ID

```
def generate_listing_id():
    conn = get_connection()
    current_list_id = list(get_data_from_db(conn, NO_SQL_QUERY["LISTING_ID"]))
    return make_it_truly_random(current_list_id)

def make_it_truly_random(current_list_id):
    random_listing_id = random.randint(111111111, 999999999)
    if random_listing_id in current_list_id:
        make_it_truly_random(current_list_id)
    return random_listing_id
```

Figure 10

➔ fetch the data from the database - designed to generate queries in the form SELECT, FROM, WHERE.

```
def get_filtered_cars_details(filter_dictionary):
   conn = get_connection()
   query = NO_SQL_QUERY.get("CAR_LISTING")
   where_logic = "where "
   for key in filter_dictionary:
            current_filter  =  FILTER_LOGIC_PATTERN.format(HTML_NAME_MAPPING_TO_COL_NAME[key],
filter_dictionary.get(key))
       if where_logic == "where ":
           where_logic = where_logic + current_filter
       else:
           where_logic = where_logic + "and " + current_filter
   if where_logic != "where ":
       formatted_query = query + where_logic
   else:
       formatted_query = query
   data_iterator = get_data_from_db(conn, formatted_query)
   return list(data_iterator)
```

Figure 11

➔ load the data to the database

```
def put_data_into_db(conn=None, js1=""):
   if conn is None:
       conn = get_connection()
   dom = conn.get_domain(DOMAIN_NAME)
   dom.batch_put_attributes(js1)
```

Figure 12

➔ form a connection with the database - we can see boto.sdb being used in order to connect with AWS.

```
def get_connection():
         conn    =    boto.sdb.connect_to_region(AWS_REGION,    aws_access_key_id=ACCESS_KEY_ID,
aws_secret_access_key=ACCESS_KEY)
   return conn
```

Figure 13

III. Database :

SimpleDB is used as the AWS Service in our app on which around 20,000 data points have been uploaded for APP Demo which can be scaled up to 3 million data points which we have stored offline.

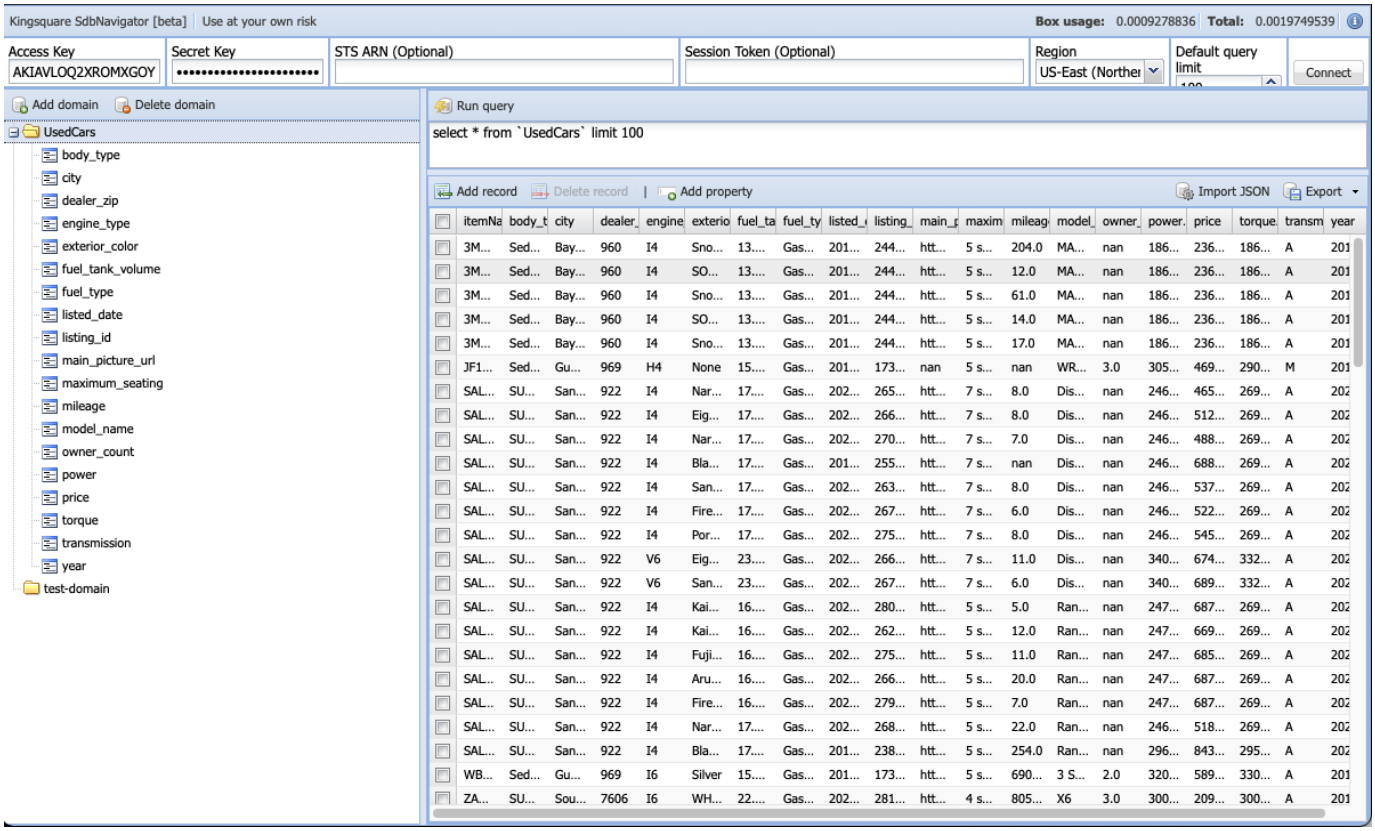


Figure 14

➔ SimpleDb data as seen using SDBNavigator before creating a Used_Cars_APP database with sample database for testing called UsedCars

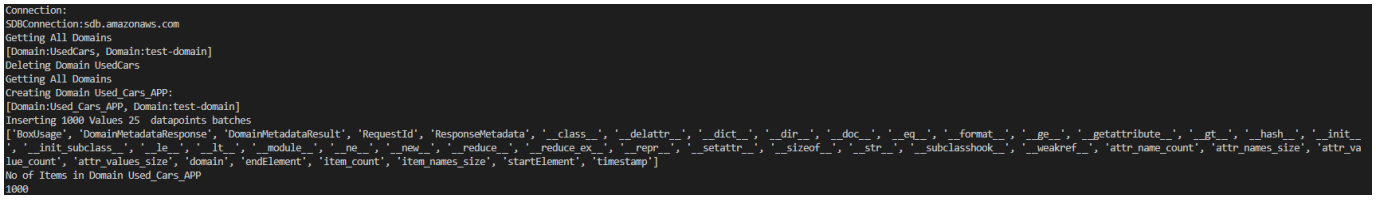


Figure 15

➔ The python console while deleting a sample domain and making a new domain for uploading data for the application

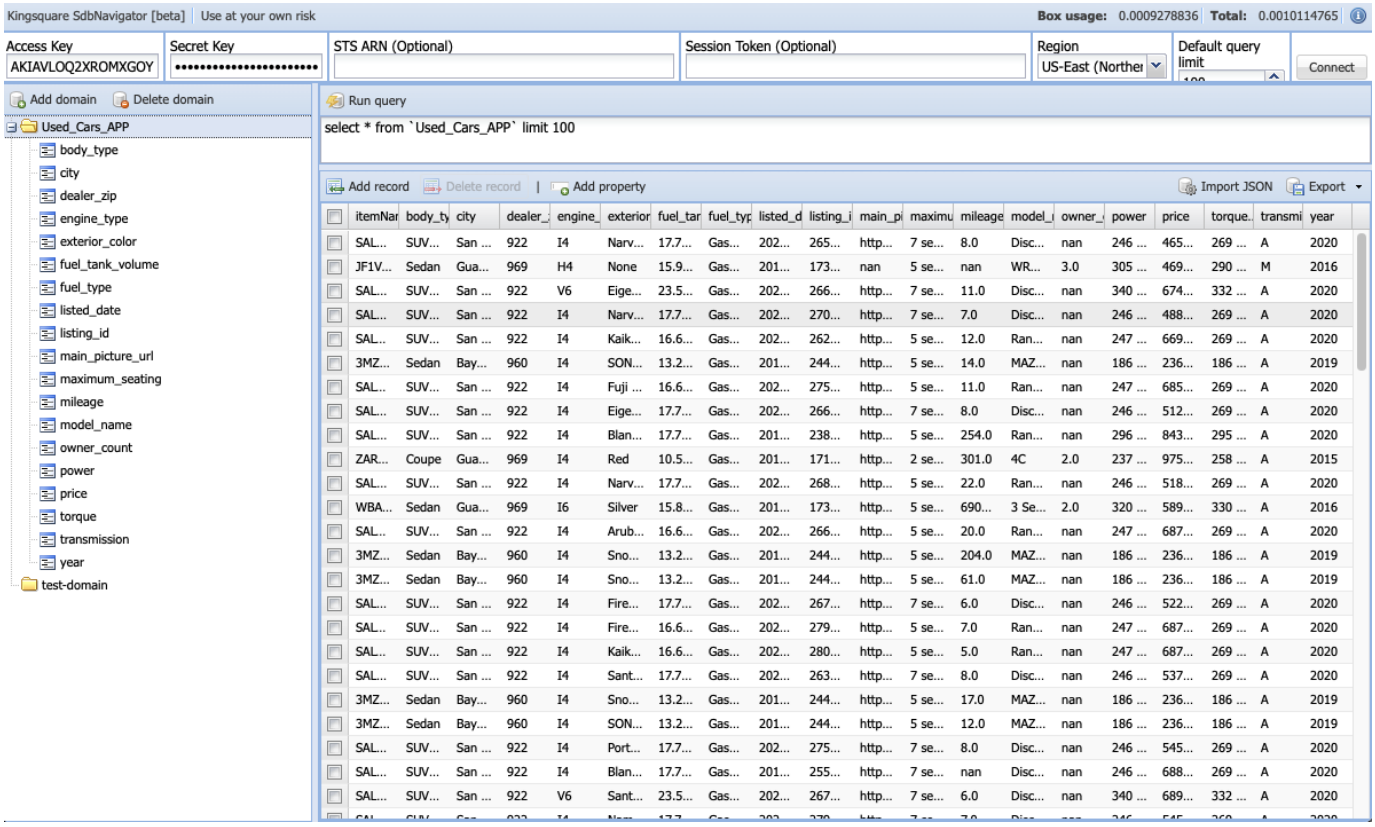


Figure 16

➔ SimpleDb data as seen using SDBNavigator after creating a Used_Cars_App database

Exporting the data into SimpleDB through boto:

To export the data into the Amazon SimpleDB we used the boto package in python to export data to our SimpleDB as mentioned above. A brief about how the data gets loaded into the database system through this.

The below code gives an idea of how the connection of simpleDb is established using boto. We can also query the data using this package in python. An example is provided in the code below to understand how a query is written and processed by the same and the output is also provided giving us the result of that query. Here the pandas library is used to read through the csv file which contains our dataset.

```
Users > swetna > Desktop > Query.py > ...
  1  import boto.sdb
  2  import pandas as pd
  3
  4  # establishing connection
  5  conn = boto.sdb.connect_to_region('us-east-1',
  6                                    aws_access_key_id='AKIAVLOQ2XROMXGOYF4L',
  7                                    aws_secret_access_key='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')
  8  print(conn)
  9
 10  # getting all domains of the used cars
 11  domain = conn.get_domain('Used_Cars_APP')
 12
 13  # To create a domain in simpleDB
 14  conn.create_domain('test-domain22')
 15
 16  # Adding data to SimpleDB via a csv file
 17  df1=pd.read_csv('/Users/swetna/Desktop/Swetna.csv')
 18  vin=df1["vin"]
 19  df1.drop('vin', inplace=True, axis=1)
 20
 21
 22  # Querying the data added to the domain
 23  # (Demo to get all cars with body_type as Coupe, exterior color matching red and the price being lower than 98000)
 24  query= 'select * from Used_Cars_APP Where body_type="Coupe" intersection exterior_color="Red" intersection price < "98000" limit 5'
 25  rs = domain.select(query)
 26  for j in rs:
 27      print(j)
 28
 29
 30  # Getting all domains and printing them
 31  domains = conn.get_all_domains()
 32  print(domains)
 33
 34  print(domains.item_count) #getting the number of domains created in SimpleDB
 35
```

Figure17.1: Code for SimpleDB

```
% /usr/local/bin/python3 /Users/swetna/Desktop/Query.py
Search (⇧⌘F)  :sdb.amazonaws.com
{'body_type': 'Coupe', 'city': 'Guaynabo', 'dealer_zip': '969', 'engine_type': 'I4', 'exterior_color': 'Red', 'fuel_tank_volume': '10.5 gal', 'fuel_type': 'Gas
oline', 'listed_date': '2017-04-06', 'listing_id': '171865107', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2017/04/18/18/55/2015_alfa_rome
o_4c-pic-5436617585836482578-152x114.jpeg', 'maximum_seating': '2 seats', 'mileage': '301.0', 'model_name': '4C', 'owner_count': '2.0', 'power': '237 hp @ 6,00
0 RPM', 'price': '97579.0', 'torque': '258 lb-ft @ 4,250 RPM', 'transmission': 'A', 'year': '2015'}
{'body_type': 'Coupe', 'city': 'Little Ferry', 'dealer_zip': '7643', 'engine_type': 'nan', 'exterior_color': 'Red', 'fuel_tank_volume': 'nan', 'fuel_type': 'na
n', 'listed_date': '2020-07-14', 'listing_id': '276497426', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2020/07/13/20/34/1978_chevrolet_el_
camino-pic-2097645105888944012-152x114.jpeg', 'maximum_seating': 'nan', 'mileage': '68064.0', 'model_name': 'El Camino', 'owner_count': 'nan', 'power': 'nan',
'price': '9250.0', 'torque': 'nan', 'transmission': 'A', 'year': '1978'}
{'body_type': 'Coupe', 'city': 'Lakewood', 'dealer_zip': '8701', 'engine_type': 'I4', 'exterior_color': 'Red', 'fuel_tank_volume': '13 gal', 'fuel_type': 'Gaso
line', 'listed_date': '2020-07-26', 'listing_id': '277650187', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2020/08/22/20/30/2003_honda_civi
c_coupe-pic-7774390226951843307-152x114.jpeg', 'maximum_seating': '5 seats', 'mileage': '175418.0', 'model_name': 'Civic Coupe', 'owner_count': '4.0', 'power':
 '127 hp @ 6,300 RPM', 'price': '1999.0', 'torque': '114 lb-ft @ 4,800 RPM', 'transmission': 'A', 'year': '2003'}
{'body_type': 'Coupe', 'city': 'Teterboro', 'dealer_zip': '7608', 'engine_type': 'nan', 'exterior_color': 'Red', 'fuel_tank_volume': '19 gal', 'fuel_type': 'na
n', 'listed_date': '2020-07-15', 'listing_id': '276650311', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2020/07/15/08/15/2010_chevrolet_cam
aro-pic-3356414226829231610-152x114.jpeg', 'maximum_seating': '4 seats', 'mileage': '32084.0', 'model_name': 'Camaro', 'owner_count': '1.0', 'power': '400 hp @
 5,900 RPM', 'price': '23999.0', 'torque': '410 lb-ft @ 4,300 RPM', 'transmission': 'A', 'year': '2010'}
{'body_type': 'Coupe', 'city': 'Milford', 'dealer_zip': '6460', 'engine_type': 'V8', 'exterior_color': 'Red', 'fuel_tank_volume': '16 gal', 'fuel_type': 'Gasol
ine', 'listed_date': '2020-07-21', 'listing_id': '277120994', 'main_picture_url': 'nan', 'maximum_seating': '4 seats', 'mileage': '6.0', 'model_name': 'Mustang
', 'owner_count': 'nan', 'power': '460 hp @ 7,000 RPM', 'price': '49105.0', 'torque': '420 lb-ft @ 4,600 RPM', 'transmission': 'A', 'year': '2020'}
{'body_type': 'Coupe', 'city': 'Newark', 'dealer_zip': '7105', 'engine_type': 'V8', 'exterior_color': 'Red', 'fuel_tank_volume': '18.5 gal', 'fuel_type': 'Gaso
line', 'listed_date': '2020-01-11', 'listing_id': '262931394', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2020/04/15/04/08/2017_dodge_chal
lenger-pic-1793869705463298828-152x114.jpeg', 'maximum_seating': '5 seats', 'mileage': '74900.0', 'model_name': 'Challenger', 'owner_count': '2.0', 'power': '7
07 hp @ 6,200 RPM', 'price': '44995.0', 'torque': '650 lb-ft @ 4,800 RPM', 'transmission': 'A', 'year': '2017'}
{'body_type': 'Coupe', 'city': 'Newark', 'dealer_zip': '7105', 'engine_type': 'V8', 'exterior_color': 'Red', 'fuel_tank_volume': '22.7 gal', 'fuel_type': 'Gaso
line', 'listed_date': '2020-06-20', 'listing_id': '274580551', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2020/07/25/08/19/2010_ferrari_45
8_italia-pic-4951158472319992975-152x114.jpeg', 'maximum_seating': '2 seats', 'mileage': '42719.0', 'model_name': '458 Italia', 'owner_count': '4.0', 'power':
'562 hp @ 9,000 RPM', 'price': '139995.0', 'torque': '398 lb-ft @ 6,000 RPM', 'transmission': 'A', 'year': '2010'}
{'body_type': 'Coupe', 'city': 'South Hackensack', 'dealer_zip': '7606', 'engine_type': 'I4', 'exterior_color': 'Red', 'fuel_tank_volume': '16 gal', 'fuel_type
': 'Gasoline', 'listed_date': '2020-08-08', 'listing_id': '278759017', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2020/08/25/19/08/2015_ca
dillac_ats_coupe-pic-6308640846832605165-152x114.jpeg', 'maximum_seating': '4 seats', 'mileage': '68775.0', 'model_name': 'ATS Coupe', 'owner_count': '3.0', 'p
ower': '272 hp @ 5,500 RPM', 'price': '15995.0', 'torque': '295 lb-ft @ 3,000 RPM', 'transmission': 'A', 'year': '2015'}
{'body_type': 'Coupe', 'city': 'New York', 'dealer_zip': '10019', 'engine_type': 'nan', 'exterior_color': 'Red', 'fuel_tank_volume': 'nan', 'fuel_type': 'nan',
 'listed_date': '2020-02-12', 'listing_id': '265574821', 'main_picture_url': 'https://static.cargurus.com/images/forsale/2020/02/11/13/41/1968_porsche_912-pic-
2191839067224895842-152x114.jpeg', 'maximum_seating': 'nan', 'mileage': 'nan', 'model_name': '912', 'owner_count': 'nan', 'power': 'nan', 'price': '105500.0',
'torque': 'nan', 'transmission': 'M', 'year': '1968'}
[Domain:Used_Cars_APP, Domain:aarti-test, Domain:test-domain22]
```

Figure 17.2: Output of the code from Figure17.1