Data Engineering

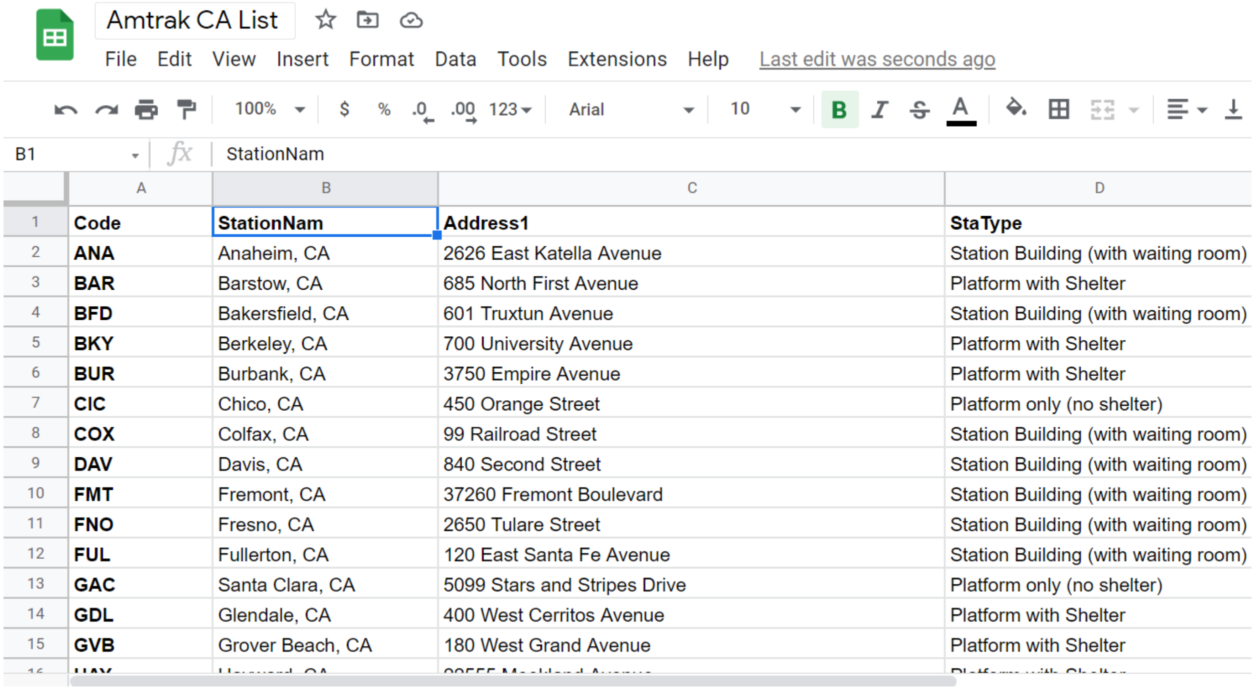
LAB 8 - Final Project - Amtrak Train Data

James Berardini

Due Date: March 13, 2022

Data

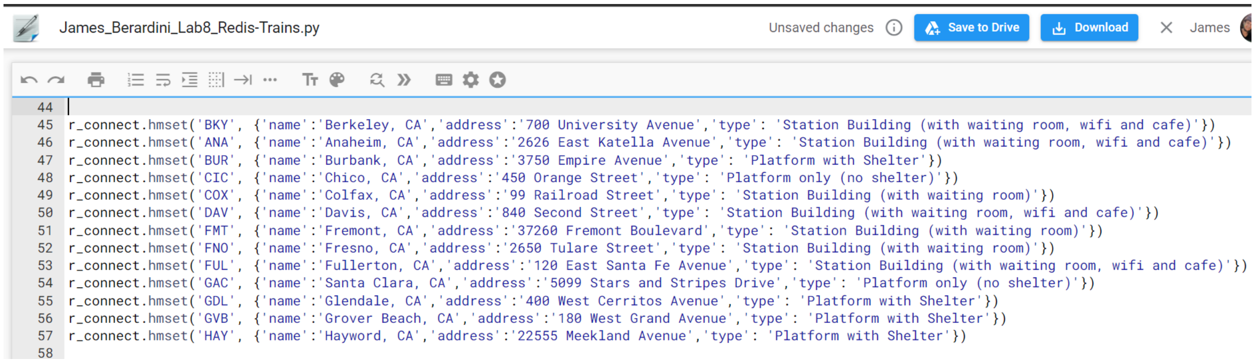
The data used for this report was a list of train stations, schedule times and train numbers. The data was formatted and prepped in sheets and then inserted through Python code. The data was used to create a Redis database.


Amtrak CA List
File Edit View Insert Format Data Tools Extensions Help
Last edit was seconds ago
100% $ % .0 .00 123 Arial 10
B1
fx StationNam

	A	B	C	D
1	Code	StationNam	Address1	StaType
2	ANA	Anaheim, CA	2626 East Katella Avenue	Station Building (with waiting room)
3	BAR	Barstow, CA	685 North First Avenue	Platform with Shelter
4	BFD	Bakersfield, CA	601 Truxtun Avenue	Station Building (with waiting room)
5	BKY	Berkeley, CA	700 University Avenue	Platform with Shelter
6	BUR	Burbank, CA	3750 Empire Avenue	Platform with Shelter
7	CIC	Chico, CA	450 Orange Street	Platform only (no shelter)
8	COX	Colfax, CA	99 Railroad Street	Station Building (with waiting room)
9	DAV	Davis, CA	840 Second Street	Station Building (with waiting room)
10	FMT	Fremont, CA	37260 Fremont Boulevard	Station Building (with waiting room)
11	FNO	Fresno, CA	2650 Tulare Street	Station Building (with waiting room)
12	FUL	Fullerton, CA	120 East Santa Fe Avenue	Station Building (with waiting room)
13	GAC	Santa Clara, CA	5099 Stars and Stripes Drive	Platform only (no shelter)
14	GDL	Glendale, CA	400 West Cerritos Avenue	Platform with Shelter
15	GVB	Grover Beach, CA	180 West Grand Avenue	Platform with Shelter
16	HAY	Hayward, CA	22555 Meekland Avenue	Platform with Shelter

Figure 1: Data for the Train Stations. The data will be stored as a hash in Redis.

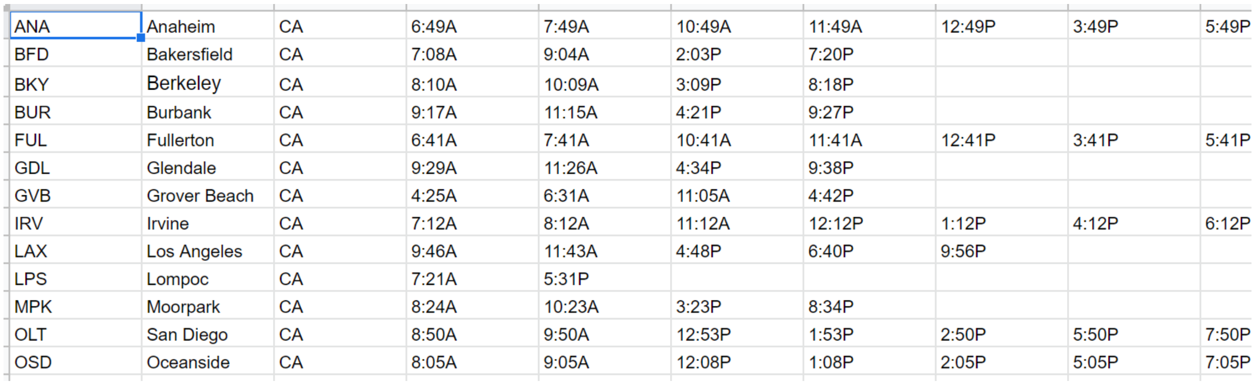
In figure 2 below, Python code was utilized for the station insert. The “hmset” command was used to create a hash table with the key being the station code.



```
r_connect.hmset('BKY', {'name':'Berkeley, CA','address':'700 University Avenue','type': 'Station Building (with waiting room, wifi and cafe)'})
r_connect.hmset('ANA', {'name':'Anaheim, CA','address':'2626 East Katella Avenue','type': 'Station Building (with waiting room, wifi and cafe)'})
r_connect.hmset('BUR', {'name':'Burbank, CA','address':'3750 Empire Avenue','type': 'Platform with Shelter'})
r_connect.hmset('CIC', {'name':'Chico, CA','address':'450 Orange Street','type': 'Platform only (no shelter)'})
r_connect.hmset('COX', {'name':'Colfax, CA','address':'99 Railroad Street','type': 'Station Building (with waiting room)'})
r_connect.hmset('DAV', {'name':'Davis, CA','address':'840 Second Street','type': 'Station Building (with waiting room, wifi and cafe)'})
r_connect.hmset('FMT', {'name':'Fremont, CA','address':'37260 Fremont Boulevard','type': 'Station Building (with waiting room)'})
r_connect.hmset('FNO', {'name':'Fresno, CA','address':'2650 Tulare Street','type': 'Station Building (with waiting room)'})
r_connect.hmset('FUL', {'name':'Fullerton, CA','address':'120 East Santa Fe Avenue','type': 'Station Building (with waiting room, wifi and cafe)'})
r_connect.hmset('GAC', {'name':'Santa Clara, CA','address':'5099 Stars and Stripes Drive','type': 'Platform only (no shelter)'})
r_connect.hmset('GDL', {'name':'Glendale, CA','address':'400 West Cerritos Avenue','type': 'Platform with Shelter'})
r_connect.hmset('GVB', {'name':'Grover Beach, CA','address':'180 West Grand Avenue','type': 'Platform with Shelter'})
r_connect.hmset('HAY', {'name':'Hayword, CA','address':'22555 Meekland Avenue','type': 'Platform with Shelter'})
```

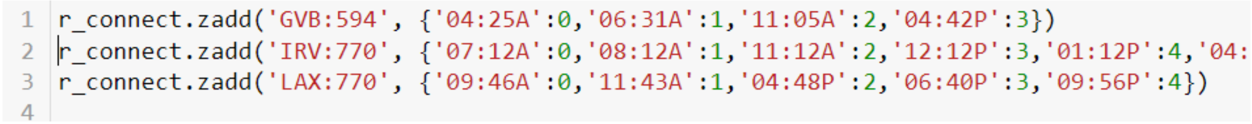
Figure 2: Sample of Python inserts for Redis hash table for the train stations.

For the departure times I found PDF files of train schedules. I copied and pasted this information for 28 different schedules and formatted them for Python code insert. This part of the project was time intensive.



ANA	Anaheim	CA	6:49A	7:49A	10:49A	11:49A	12:49P	3:49P	5:49P
BFD	Bakersfield	CA	7:08A	9:04A	2:03P	7:20P			
BKY	Berkeley	CA	8:10A	10:09A	3:09P	8:18P			
BUR	Burbank	CA	9:17A	11:15A	4:21P	9:27P			
FUL	Fullerton	CA	6:41A	7:41A	10:41A	11:41A	12:41P	3:41P	5:41P
GDL	Glendale	CA	9:29A	11:26A	4:34P	9:38P			
GVB	Grover Beach	CA	4:25A	6:31A	11:05A	4:42P			
IRV	Irvine	CA	7:12A	8:12A	11:12A	12:12P	1:12P	4:12P	6:12P
LAX	Los Angeles	CA	9:46A	11:43A	4:48P	6:40P	9:56P		
LPS	Lompoc	CA	7:21A	5:31P					
MPK	Moorpark	CA	8:24A	10:23A	3:23P	8:34P			
OLT	San Diego	CA	8:50A	9:50A	12:53P	1:53P	2:50P	5:50P	7:50P
OSD	Oceanside	CA	8:05A	9:05A	12:08P	1:08P	2:05P	5:05P	7:05P

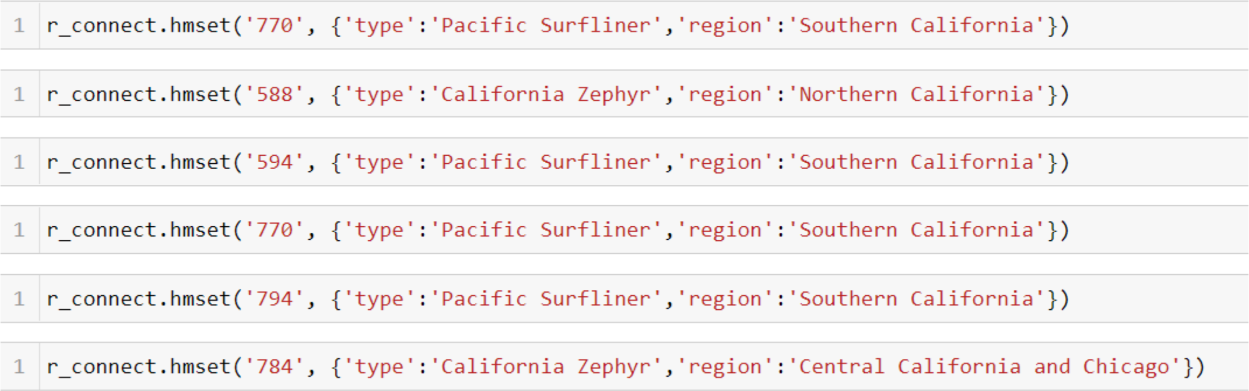
Figure 3: Sheets data for train schedule insert.



```
r_connect.zadd('GVB:594', {'04:25A':0,'06:31A':1,'11:05A':2,'04:42P':3})
r_connect.zadd('IRV:770', {'07:12A':0,'08:12A':1,'11:12A':2,'12:12P':3,'01:12P':4,'04:
r_connect.zadd('LAX:770', {'09:46A':0,'11:43A':1,'04:48P':2,'06:40P':3,'09:56P':4})
```

Figure 4: Formatted schedule times for Python. Data was inserted into a sorted Redis set.

The final data was finding train numbers on the PDF sheet. Most of the train numbers were assigned arbitrarily to different stations for proof of concept.



```
r_connect.hmset('770', {'type':'Pacific Surfliner','region':'Southern California'})
```

```
r_connect.hmset('588', {'type':'California Zephyr','region':'Northern California'})
```

```
r_connect.hmset('594', {'type':'Pacific Surfliner','region':'Southern California'})
```

```
r_connect.hmset('770', {'type':'Pacific Surfliner','region':'Southern California'})
```

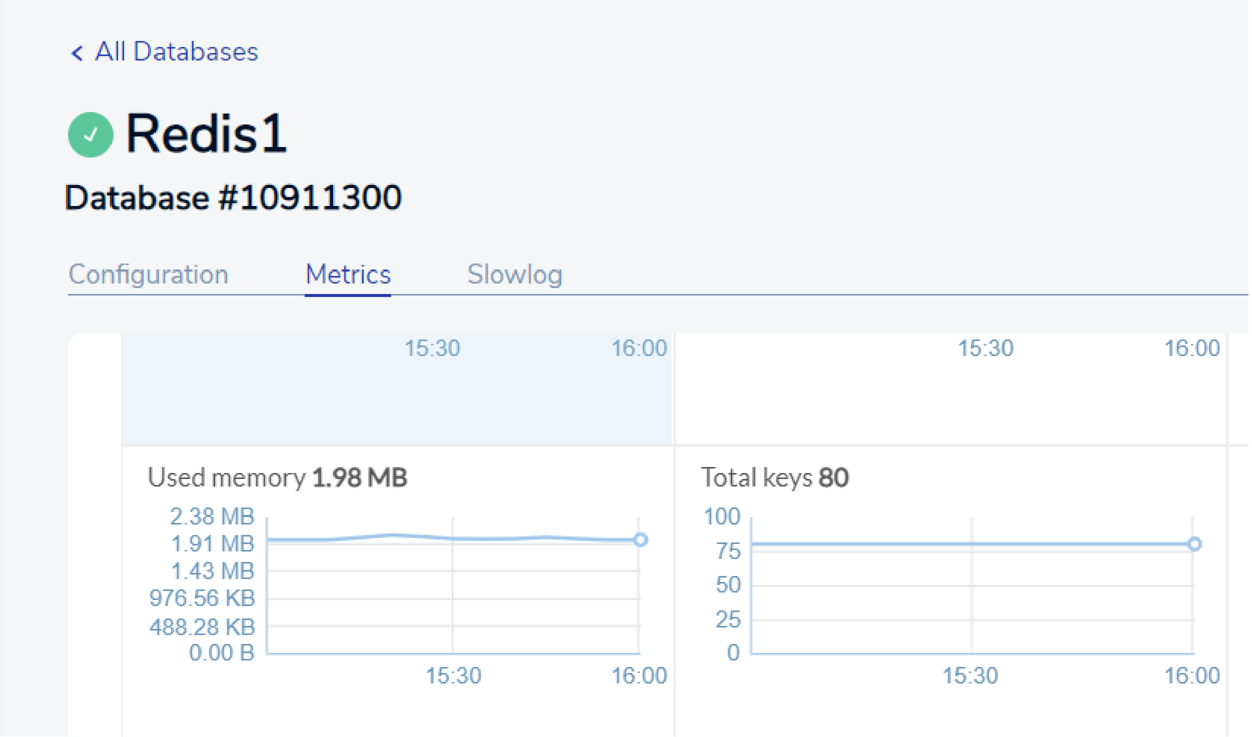
```
r_connect.hmset('794', {'type':'Pacific Surfliner','region':'Southern California'})
```

```
r_connect.hmset('784', {'type':'California Zephyr','region':'Central California and Chicago'})
```

Figure 5: Train numbers with the type and region of trains.

Database

I created a redislabs.com free 25mb database to store the data.



Redis1

Database #10911300

Used memory 1.98 MB

Total keys 80

Figure 6: Redis database from redislabs.com. 80 keys were created using about 2mb in memory.

The database has a hash table for the stations. The key is the station code with each code storing the name, address and type of station.

The trains are stored with the train number as the key with the type and region of the train.

The departure times are stored with the station code and the train number in a concatenated key. The data stored for each key is a sorted set of departure times.

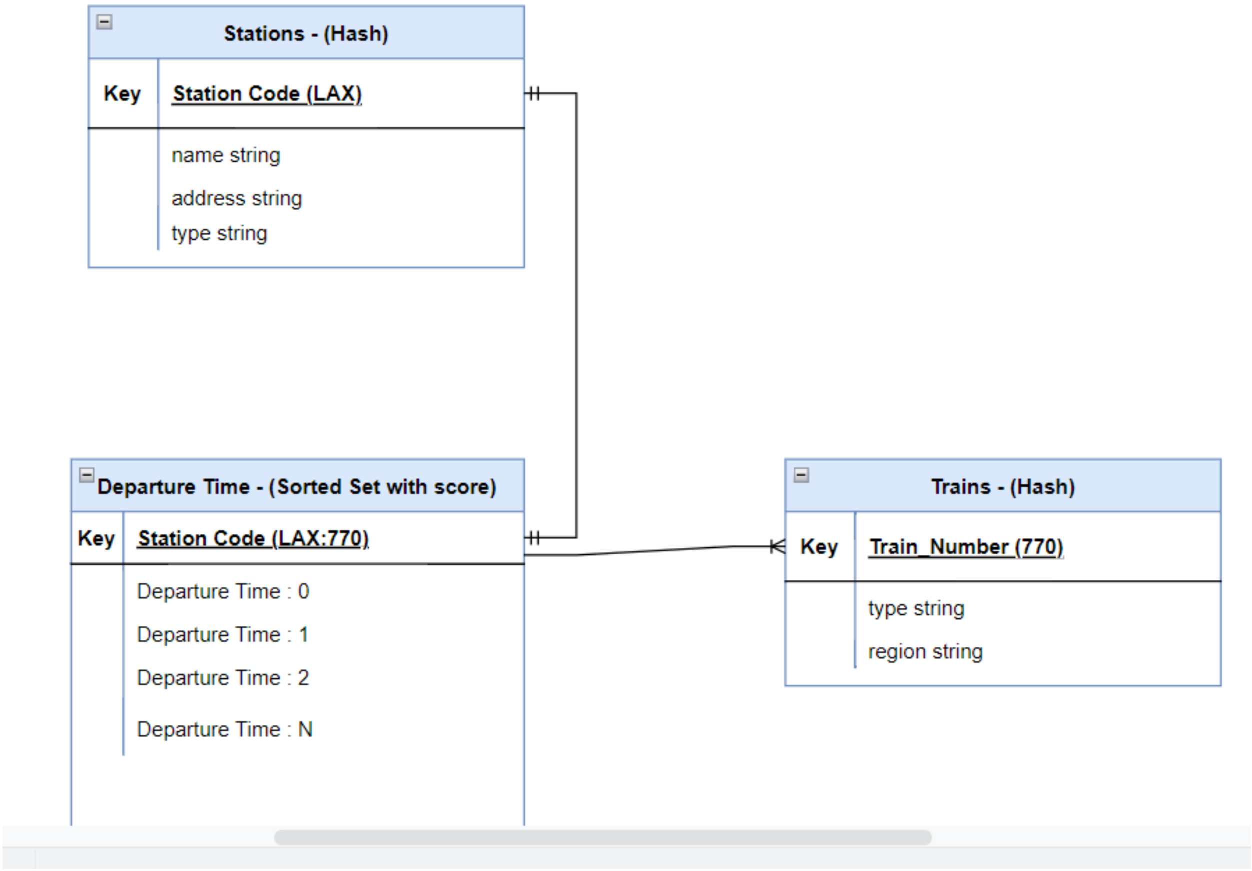
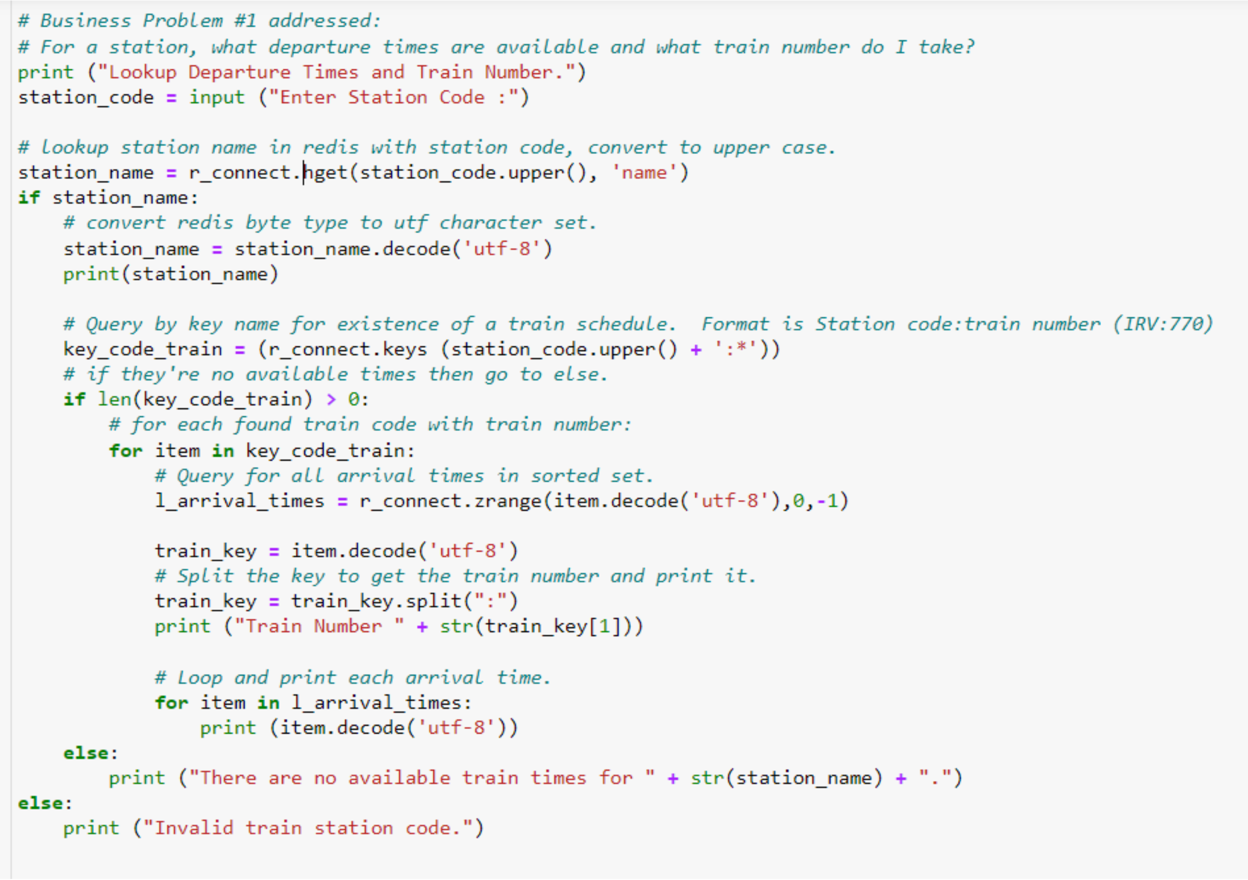


Figure 7: The Entity Relationship Diagram with 3 sets of data. Stations, Train Numbers and Departure Times.

Business Problems

Four business problems were identified to solve. Python code was written for each of these to query the Redis database.

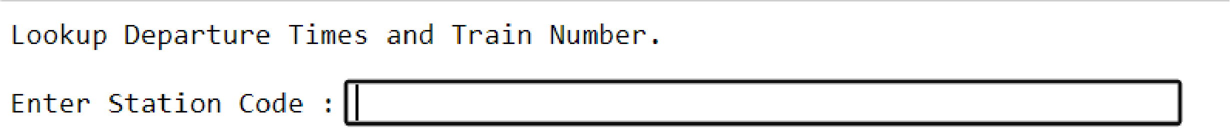
1. Business Problem: For a station, what departure times are available and what train number does a passenger take?



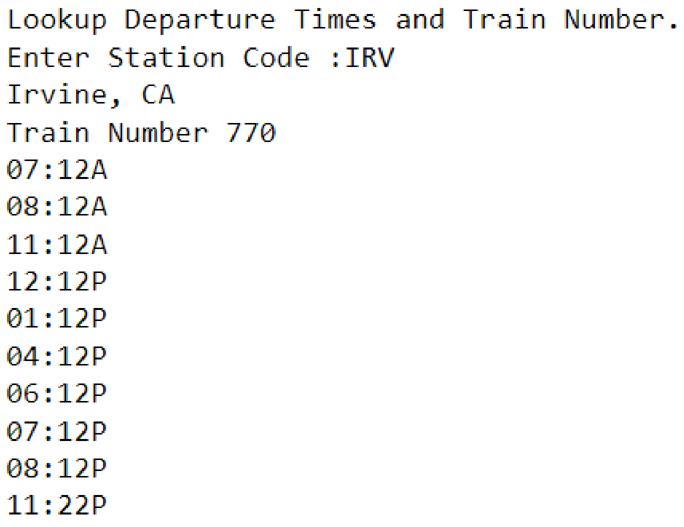
```
# Business Problem #1 addressed:
# For a station, what departure times are available and what train number do I take?
print ("Lookup Departure Times and Train Number.")
station_code = input ("Enter Station Code :")

# lookup station name in redis with station code, convert to upper case.
station_name = r_connect.hget(station_code.upper(), 'name')
if station_name:
    # convert redis byte type to utf character set.
    station_name = station_name.decode('utf-8')
    print(station_name)

    # Query by key name for existence of a train schedule.  Format is Station code:train number (IRV:770)
    key_code_train = (r_connect.keys (station_code.upper() + ':*'))
    # if they're no available times then go to else.
    if len(key_code_train) > 0:
        # for each found train code with train number:
        for item in key_code_train:
            # Query for all arrival times in sorted set.
            l_arrival_times = r_connect.zrange(item.decode('utf-8'),0,-1)

            train_key = item.decode('utf-8')
            # Split the key to get the train number and print it.
            train_key = train_key.split(":")
            print ("Train Number " + str(train_key[1]))

            # Loop and print each arrival time.
            for item in l_arrival_times:
                print (item.decode('utf-8'))
    else:
        print ("There are no available train times for " + str(station_name) + ".")
else:
    print ("Invalid train station code.")
```

Code Sample 1: For finding departure times by traincode.

The passenger enters a station code with the resulting train number and departure times listed:

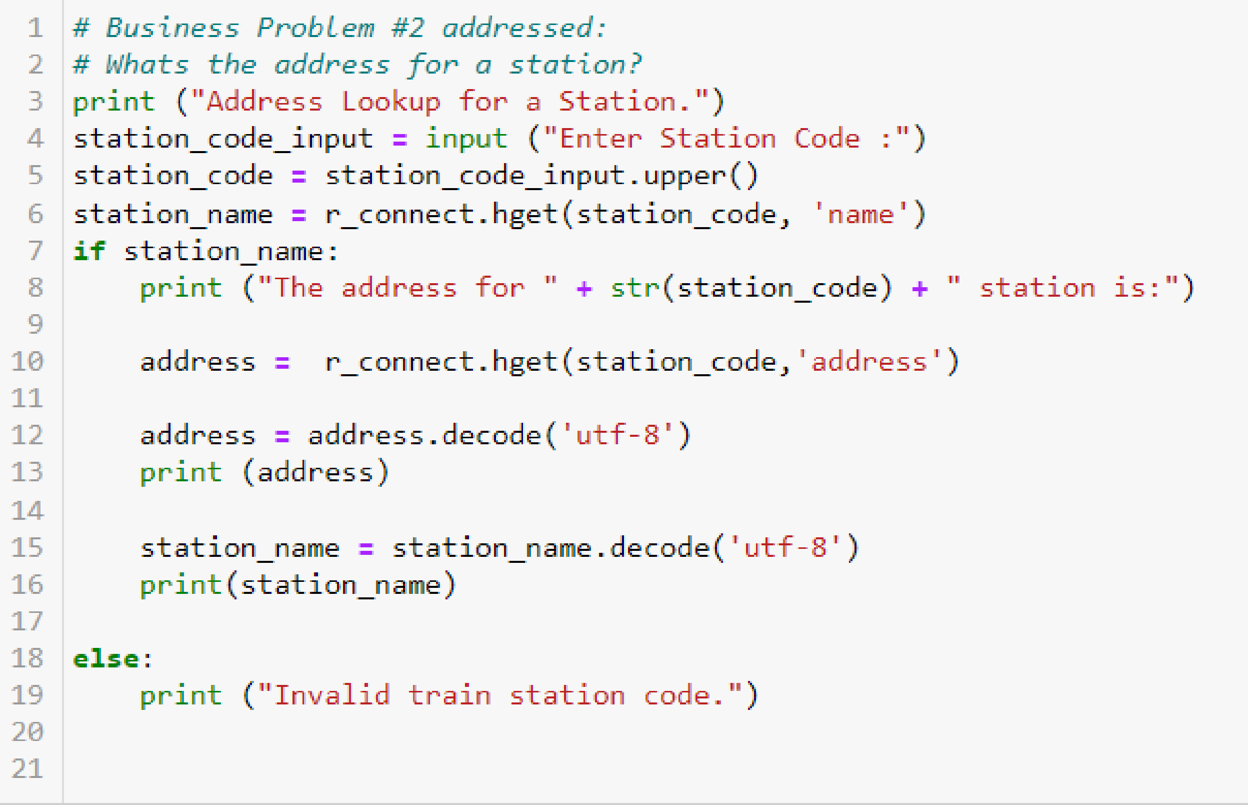


```
Lookup Departure Times and Train Number.

Enter Station Code :
```



```
Lookup Departure Times and Train Number.
Enter Station Code :IRV
Irvine, CA
Train Number 770
07:12A
08:12A
11:12A
12:12P
01:12P
04:12P
06:12P
07:12P
08:12P
11:22P
```

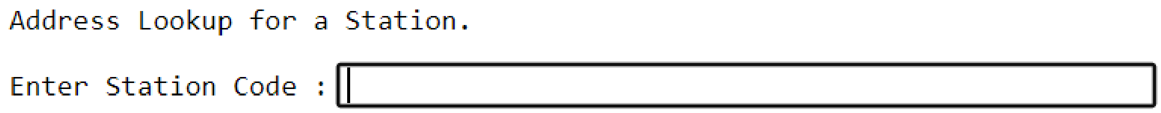
2. Business Problem: What’s the address for a station?



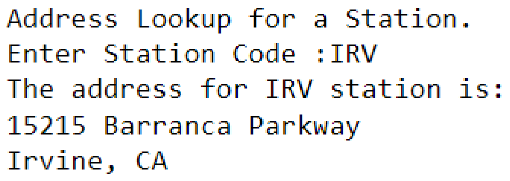
```python
# Business Problem #2 addressed:
# Whats the address for a station?
print ("Address Lookup for a Station.")
station_code_input = input ("Enter Station Code :")
station_code = station_code_input.upper()
station_name = r_connect.hget(station_code, 'name')
if station_name:
    print ("The address for " + str(station_code) + " station is:")

    address =  r_connect.hget(station_code,'address')

    address = address.decode('utf-8')
    print (address)

    station_name = station_name.decode('utf-8')
    print(station_name)

else:
    print ("Invalid train station code.")
```

Code Sample 2: Inquiring for an address by station code.

The station code is entered to display the address of the desired station.

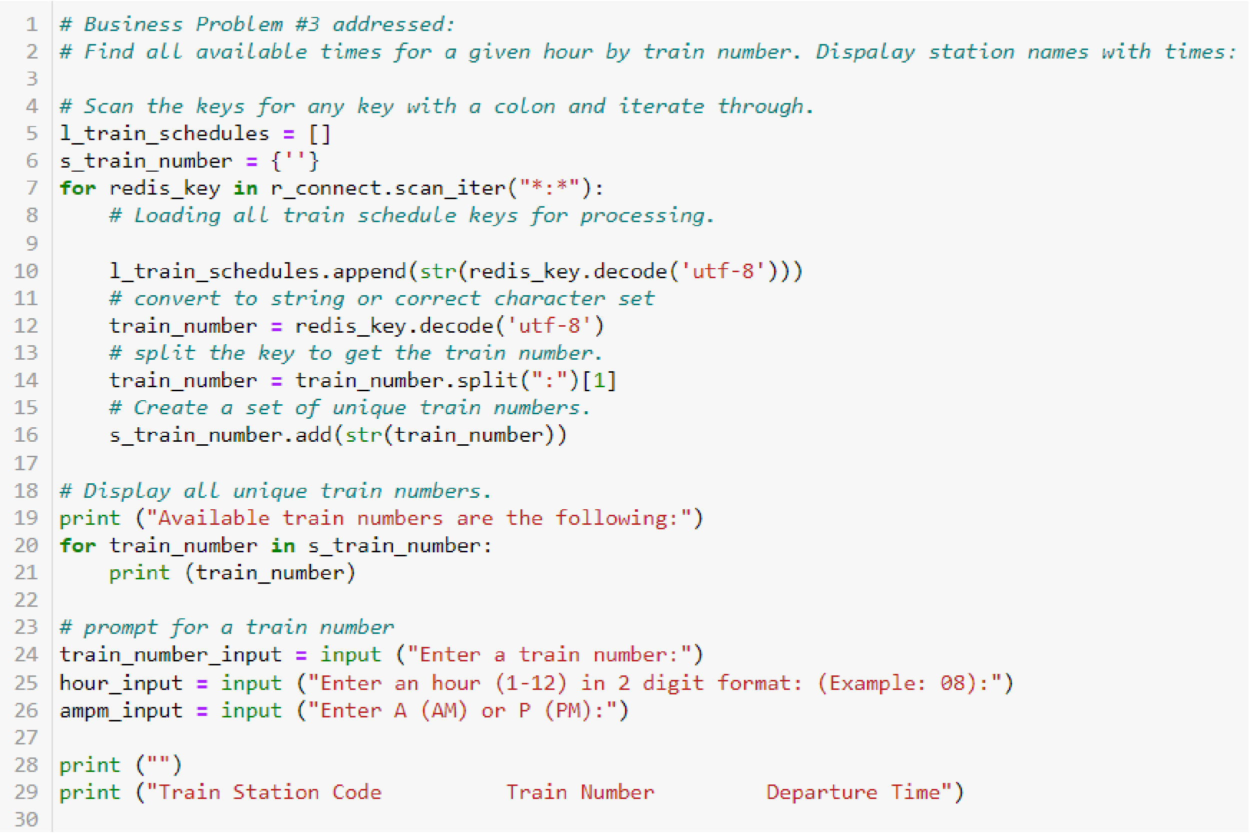


```
Address Lookup for a Station.

Enter Station Code :
```

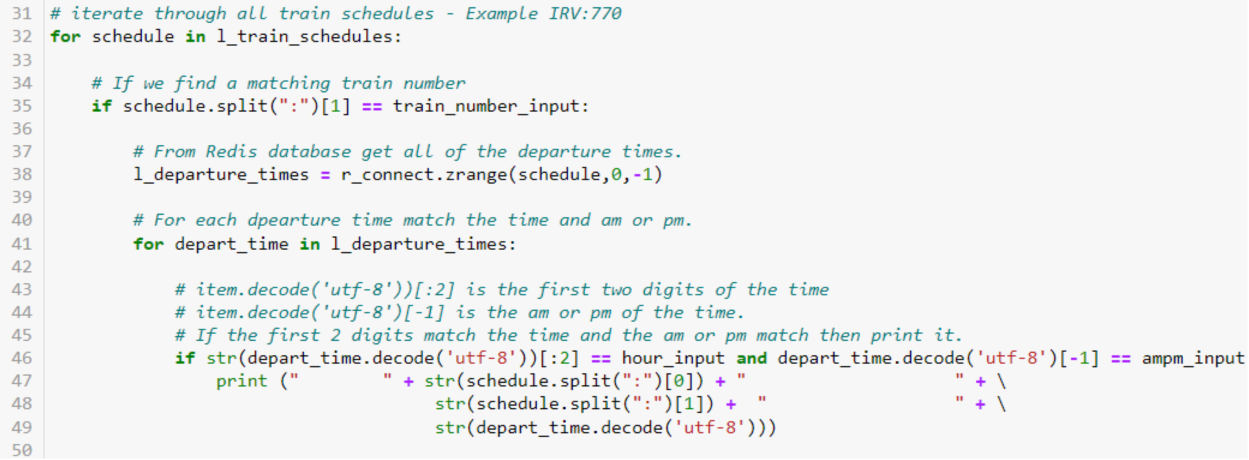


```
Address Lookup for a Station.
Enter Station Code :IRV
The address for IRV station is:
15215 Barranca Parkway
Irvine, CA
```

3. Business Problem: Find all available times for a given hour by train number. Display station names with times.



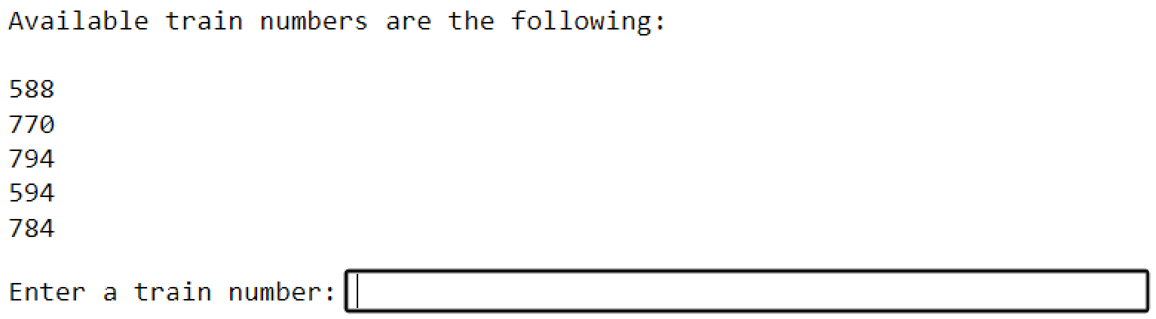
```
# Business Problem #3 addressed:
# Find all available times for a given hour by train number. Dispalay station names with times:

# Scan the keys for any key with a colon and iterate through.
l_train_schedules = []
s_train_number = {''}
for redis_key in r_connect.scan_iter("*:*"):
    # Loading all train schedule keys for processing.

    l_train_schedules.append(str(redis_key.decode('utf-8')))
    # convert to string or correct character set
    train_number = redis_key.decode('utf-8')
    # split the key to get the train number.
    train_number = train_number.split(":")[1]
    # Create a set of unique train numbers.
    s_train_number.add(str(train_number))

# Display all unique train numbers.
print ("Available train numbers are the following:")
for train_number in s_train_number:
    print (train_number)

# prompt for a train number
train_number_input = input ("Enter a train number:")
hour_input = input ("Enter an hour (1-12) in 2 digit format: (Example: 08):")
ampm_input = input ("Enter A (AM) or P (PM):")

print ("")
print ("Train Station Code          Train Number         Departure Time")

```



```python
# iterate through all train schedules - Example IRV:770
for schedule in l_train_schedules:

    # If we find a matching train number
    if schedule.split(":")[1] == train_number_input:

        # From Redis database get all of the departure times.
        l_departure_times = r_connect.zrange(schedule,0,-1)

        # For each dpearture time match the time and am or pm.
        for depart_time in l_departure_times:

            # item.decode('utf-8'))[:2] is the first two digits of the time
            # item.decode('utf-8')[-1] is the am or pm of the time.
            # If the first 2 digits match the time and the am or pm match then print it.
            if str(depart_time.decode('utf-8'))[:2] == hour_input and depart_time.decode('utf-8')[-1] == ampm_input
                print ("        " + str(schedule.split(":")[0]) + "                    " + \
                                     str(schedule.split(":")[1]) +  "                  " + \
                                     str(depart_time.decode('utf-8')))

```

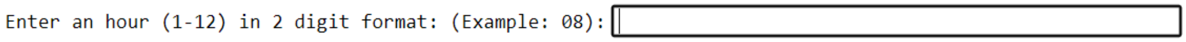
Sample Code 3: List all stations and departure times for a given hour.

The available train numbers are displayed. The passenger enters a train number



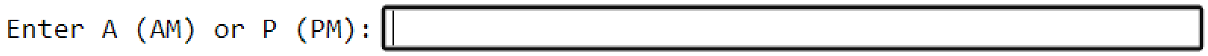
```
Available train numbers are the following:

588
770
794
594
784

Enter a train number:
```

The hour is prompted for.



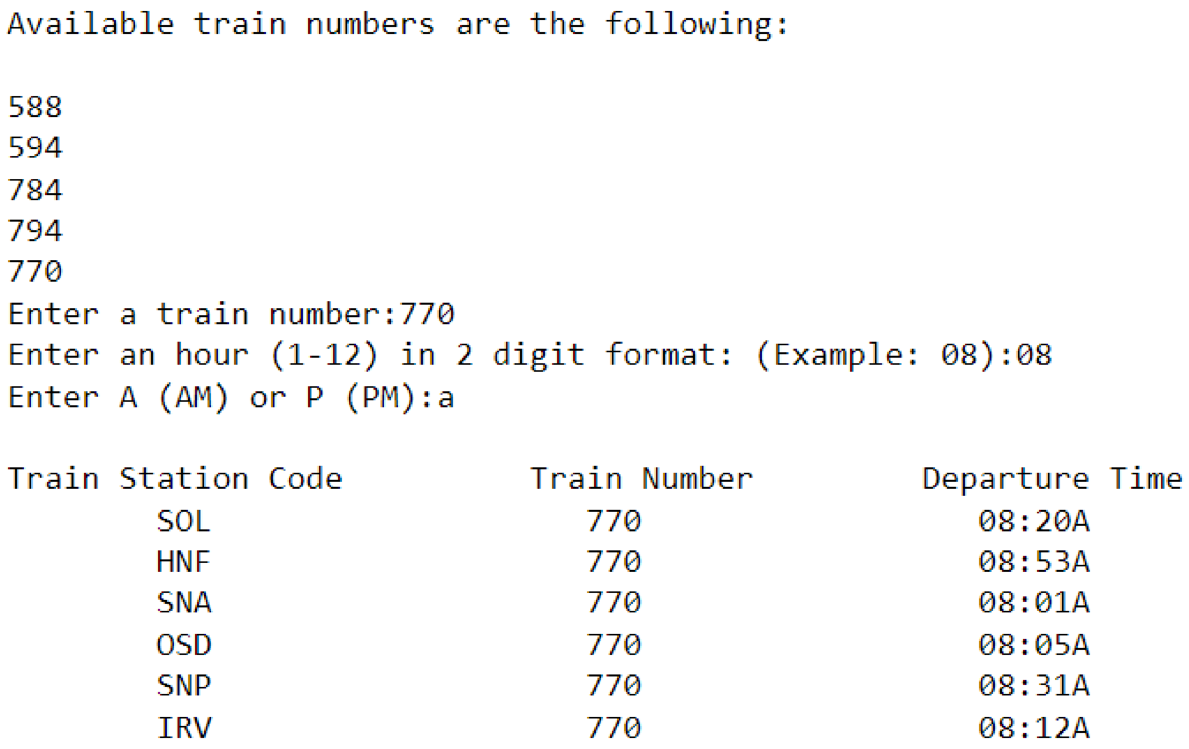
```
Enter an hour (1-12) in 2 digit format: (Example: 08):
```

Also, Am or Pm is prompted:



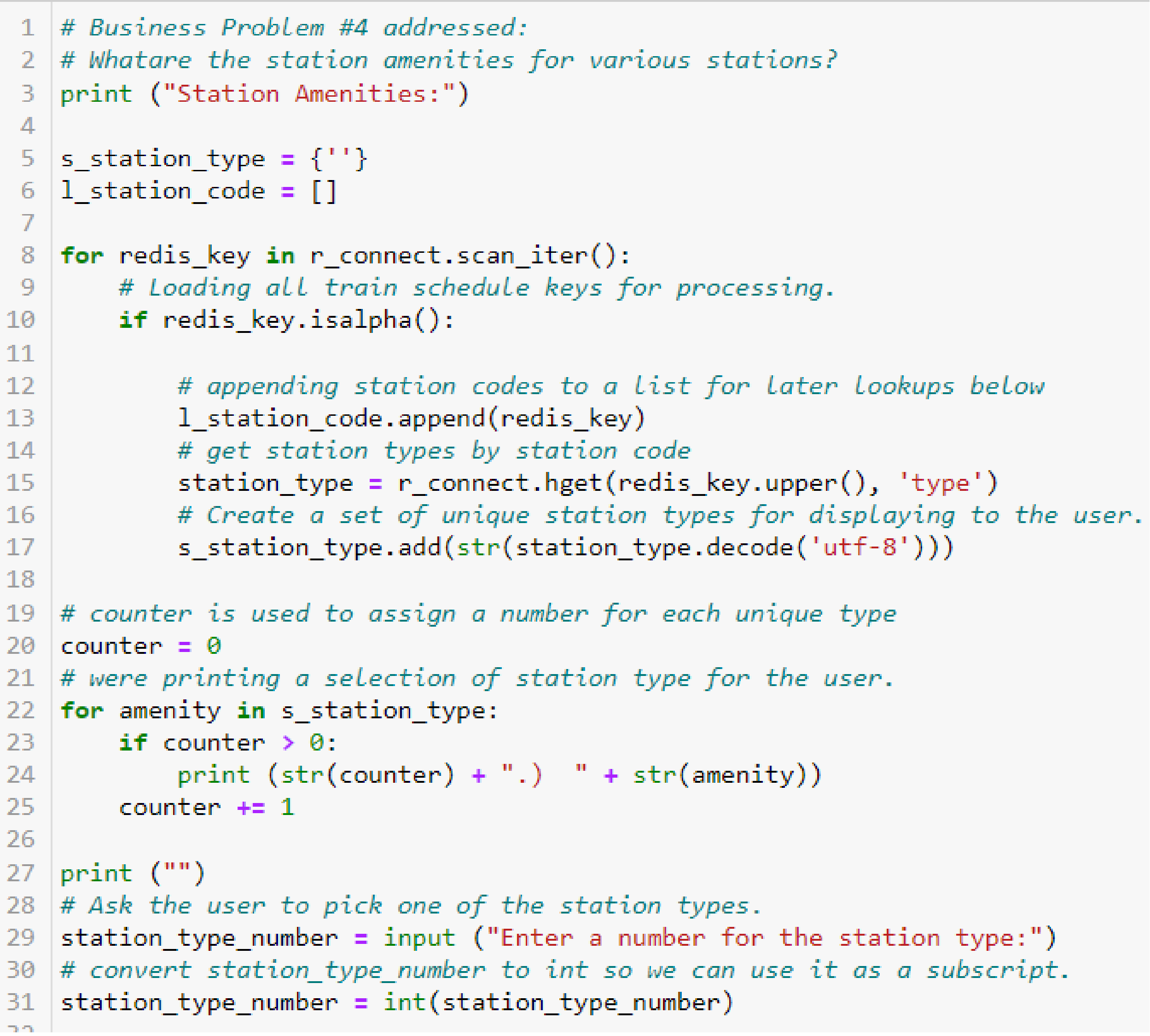
```
Enter A (AM) or P (PM):
```

The results show all station codes, train numbers departing during the 8am hour.

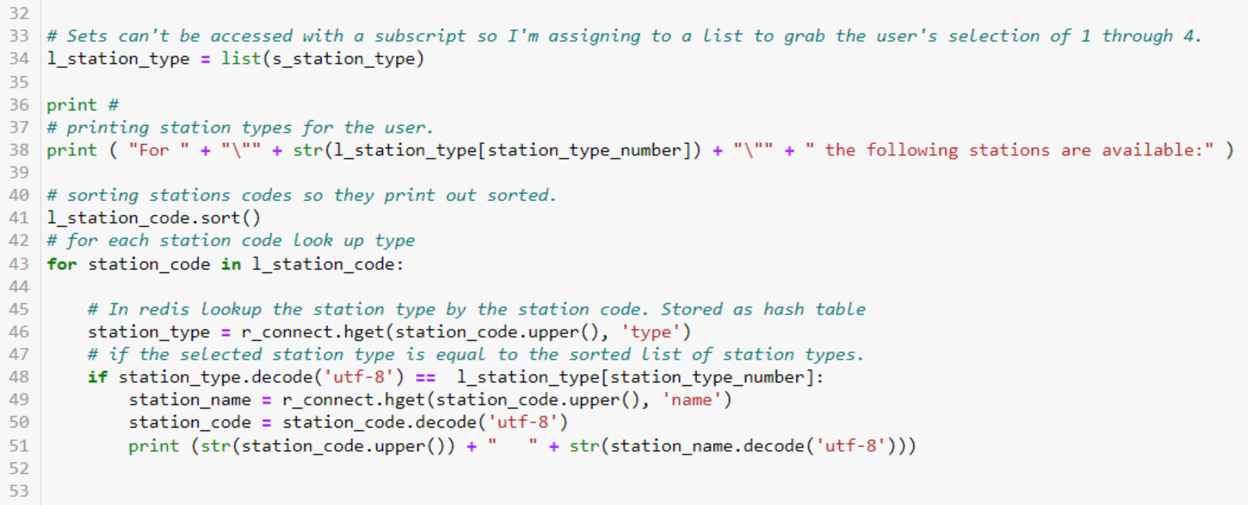


```
Available train numbers are the following:

588
594
784
794
770
Enter a train number:770
Enter an hour (1-12) in 2 digit format: (Example: 08):08
Enter A (AM) or P (PM):a

Train Station Code          Train Number          Departure Time
        SOL                     770                   08:20A
        HNF                     770                   08:53A
        SNA                     770                   08:01A
        OSD                     770                   08:05A
        SNP                     770                   08:31A
        IRV                     770                   08:12A
```

4. Business Problem: What are the station amenities for various stations?



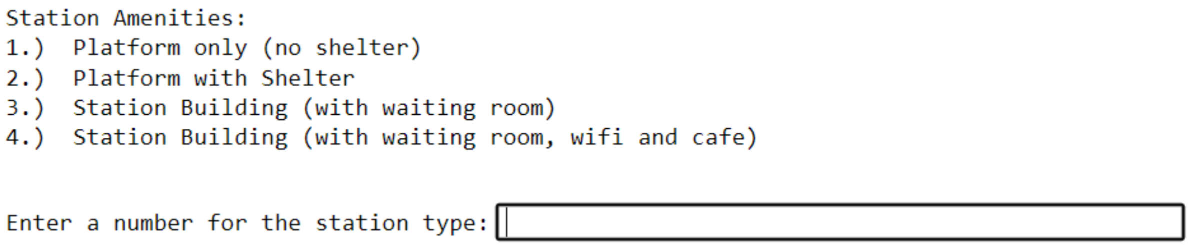
```python
# Business Problem #4 addressed:
# Whatare the station amenities for various stations?
print ("Station Amenities:")

s_station_type = {''}
l_station_code = []

for redis_key in r_connect.scan_iter():
    # Loading all train schedule keys for processing.
    if redis_key.isalpha():

        # appending station codes to a list for later lookups below
        l_station_code.append(redis_key)
        # get station types by station code
        station_type = r_connect.hget(redis_key.upper(), 'type')
        # Create a set of unique station types for displaying to the user.
        s_station_type.add(str(station_type.decode('utf-8')))

# counter is used to assign a number for each unique type
counter = 0
# were printing a selection of station type for the user.
for amenity in s_station_type:
    if counter > 0:
        print (str(counter) + ".)   " + str(amenity))
    counter += 1

print ("")
# Ask the user to pick one of the station types.
station_type_number = input ("Enter a number for the station type:")
# convert station_type_number to int so we can use it as a subscript.
station_type_number = int(station_type_number)
```



```python

# Sets can't be accessed with a subscript so I'm assigning to a list to grab the user's selection of 1 through 4.
l_station_type = list(s_station_type)

print #
# printing station types for the user.
print ( "For " + "\"" + str(l_station_type[station_type_number]) + "\"" + " the following stations are available:" )

# sorting stations codes so they print out sorted.
l_station_code.sort()
# for each station code look up type
for station_code in l_station_code:

    # In redis lookup the station type by the station code. Stored as hash table
    station_type = r_connect.hget(station_code.upper(), 'type')
    # if the selected station type is equal to the sorted list of station types.
    if station_type.decode('utf-8') ==  l_station_type[station_type_number]:
        station_name = r_connect.hget(station_code.upper(), 'name')
        station_code = station_code.decode('utf-8')
        print (str(station_code.upper()) + "   " + str(station_name.decode('utf-8')))


```

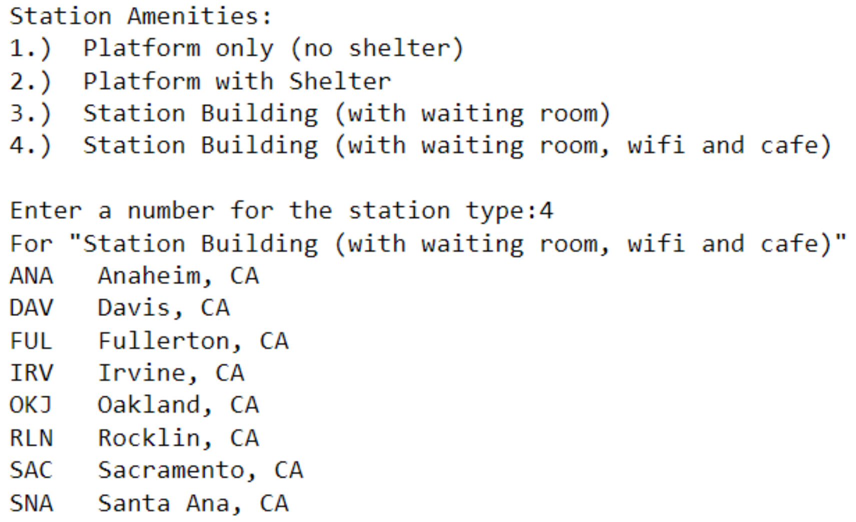
Code Sample 4: Inquiring amenities for train stations.

The passenger can query for different station amenities. In this example there are 4 types of stations.



```
Station Amenities:
1.)  Platform only (no shelter)
2.)  Platform with Shelter
3.)  Station Building (with waiting room)
4.)  Station Building (with waiting room, wifi and cafe)

Enter a number for the station type:
```

The result in this example are all stations with a waiting room, wifi and cafe.



```
Station Amenities:
1.)  Platform only (no shelter)
2.)  Platform with Shelter
3.)  Station Building (with waiting room)
4.)  Station Building (with waiting room, wifi and cafe)

Enter a number for the station type:4
For "Station Building (with waiting room, wifi and cafe)"
ANA   Anaheim, CA
DAV   Davis, CA
FUL   Fullerton, CA
IRV   Irvine, CA
OKJ   Oakland, CA
RLN   Rocklin, CA
SAC   Sacramento, CA
SNA   Santa Ana, CA
```