# HW3 - SQL

This homework has you working with a new database of information on ticket sales for various types of events. Your job will be to do some initial exploring and then demonstrate your ability to do all the different types of SQL queries we learned over the last two week. You'll also need to make one function that'll make looking at the tables easier.

These questions are written in the way someone would ask them to you. In other words, I'm using 'plain english' questions vs. ones where I'm very explicit in terms of what columns and tables to use. Your exploring of the database and functions to ease that process will come in handy here!

The database has been created using a set of data from Amazon. You can read more about what each table contains here: https://docs.aws.amazon.com/redshift/latest/dg/c_sampledb.html.

## Libraries and import functions

First bring the libraries we'll need!

```
import psycopg2
import pandas as pd
```

Now bring in all our functions we used in the lessons!

```
# Make our connection/cursor function
AWS_host_name = "ticketsdb.chzr8692xwjt.us-east-2.rds.amazonaws.com"
AWS_dbname = "ticketsdb"
AWS_user_name = "postgres"
AWS_password = "ista322ticketsdb"

def get_conn_cur(): # define function name and arguments (there aren't any)
  # Make a connection
  conn = psycopg2.connect(
    host=AWS_host_name,
    database=AWS_dbname,
    user=AWS_user_name,
    password=AWS_password,
    port='5432')

  cur = conn.cursor()   # Make a cursor after
```

✓ 0s completed at 10:15 AM

```
# Same run_query function
def run_query(query_string):

  conn, cur = get_conn_cur() # get connection and cursor

  cur.execute(query_string) # executing string as before

  my_data = cur.fetchall() # fetch query data as before

  # here we're extracting the 0th element for each item in cur.description
  colnames = [desc[0] for desc in cur.description]

  cur.close() # close
  conn.close() # close

  return(colnames, my_data) # return column names AND data

# Column name function for checking out what's in a table
def get_column_names(table_name): # arguement of table_name
  conn, cur = get_conn_cur() # get connection and cursor

  # Now select column names while inserting the table name into the WERE
  column_name_query =  """SELECT column_name FROM information_schema.columns
       WHERE table_name = '%s' """ %table_name

  cur.execute(column_name_query) # exectue
  my_data = cur.fetchall() # store

  cur.close() # close
  conn.close() # close

  return(my_data) # return

# Check table_names
def get_table_names():
  conn, cur = get_conn_cur() # get connection and cursor

  # query to get table names
  table_name_query = """SELECT table_name FROM information_schema.tables
       WHERE table_schema = 'public' """

  cur.execute(table_name_query) # execute
  my_data = cur.fetchall() # fetch results

  cur.close() #close cursor
  conn.close() # close connection

  return(my_data) # return your fetched results
```

## Make a SQL head function - 5 point

Make function to get the pandas equivalent of `.head()`

This function should be called `sql_head` and take a single argument of `table_name` where you specify the table name you want the head information from. It should return the column names along with the first five rows of the table along.

**For full points, return a pandas dataframe with this information so it displays nicely :)**

```
# make sql_head function
def sql_head(table_name):
  # columns = get_column_names(table_name)
  limit = 5
  sql = f"select * from {table_name} limit {limit}"
  results = run_query(sql)
  columns = results[0]
  data = results[1]
  df = pd.DataFrame(columns=columns, data=data)
  return df
```

```
# Check that it works!
sql_head(table_name = 'sales')
```

| | salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid | comm |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 36861 | 21191 | 7872 | 1875 | 4 | 728.00 | |
| **1** | 2 | 4 | 8117 | 11498 | 4337 | 1983 | 2 | 76.00 | |
| **2** | 3 | 5 | 1616 | 17433 | 8647 | 1983 | 2 | 350.00 | |

## Explore and SELECT - 5 point

Let's start this homework with some basic queries to get a look at what's in the various tables. I want you to do the following.

- Use your get_table_names() function to see what tables are in the database.
- Use your get_column_names() to get the column names of each of the tables. **Do this all within a single cell to keep it neat**.
- Make and run a query that selects all columns from the event table. Only return the first 5

rows.

- Use the `sql_head()` function you created to get the first five rows of the sales table.

```
# Getting table names
table_info = []
table_names = get_table_names()
table_names
```

```
[('users',),
 ('venue',),
 ('category',),
 ('date',),
 ('event',),
 ('listing',),
 ('sales',)]
```

```
# Getting column names
table_info = []
for tn in table_names:
  columns = get_column_names(tn)
  table_info.append([tn, columns])

for row in table_info:
  print(row)
```

```
[('users',), [('userid',), ('username',), ('firstname',), ('lastname',), ('ci
[('venue',), [('venueid',), ('venuename',), ('venuecity',), ('venuestate',),
[('category',), [('catid',), ('catgroup',), ('catname',), ('catdesc',)]]
[('date',), [('dateid',), ('caldate',), ('day',), ('week',), ('month',), ('qt
[('event',), [('eventid',), ('venueid',), ('catid',), ('dateid',), ('eventnam
[('listing',), [('listid',), ('sellerid',), ('eventid',), ('dateid',), ('numt
[('sales',), [('salesid',), ('listid',), ('sellerid',), ('buyerid',), ('event
```

```
# Could also just use a list comprehension vs a bunch of print statements :)
names = get_table_names()
[get_column_names(table_name= x) for x in names]
```

```
[[('userid',),
  ('username',),
  ('firstname',),
  ('lastname',),
  ('city',),
  ('state',),
  ('email',),
  ('phone',),
  ('likesports',),
  ('liketheatre',),
  ('likeconcerts',),
  ('likejazz',),
  ('likeclassical',),
  ('likeopera',),
  ('likerock',),
  ('likevegas',),
  ('likebroadway',),
  ('likemusicals',)],
 [('venueid',),
  ('venuename',),
  ('venuecity',),
  ('venuestate',),
  ('venueseats',)],
 [('catid',), ('catgroup',), ('catname',), ('catdesc',)],
 [('dateid',),
  ('caldate',),
  ('day',),
  ('week',),
  ('month',),
  ('qtr',),
  ('year',),
  ('holiday',)],
 [('eventid',),
  ('venueid',),
  ('catid',),
  ('dateid',),
  ('eventname',),
  ('starttime',)],
 [('listid',),
  ('sellerid',),
  ('eventid',),
  ('dateid',),
  ('numtickets',),
  ('priceperticket',),
  ('totalprice',),
  ('listtime',)],
 [('salesid',),
  ('listid',),
  ('sellerid',),
  ('buyerid',),
  ('eventid',),
  ('dateid',),
  ('qtysold',),
  ('pricepaid',),
  ('commission',),
  ('saletime',)]]
```

```
# Query on events
sq = """select * from event limit 5"""
run_query(sq)
```

```
([ 'eventid', 'venueid', 'catid', 'dateid', 'eventname', 'starttime'],
 [(1, 305, 8, 1851, 'Gotterdammerung', datetime.datetime(2008, 1, 25, 14,
30)),
  (2, 306, 8, 2114, 'Boris Godunov', datetime.datetime(2008, 10, 15, 20, 0)),
  (3, 302, 8, 1935, 'Salome', datetime.datetime(2008, 4, 19, 14, 30)),
  (4,
   309,
   8,
   2090,
   'La Cenerentola (Cinderella)',
   datetime.datetime(2008, 9, 21, 14, 30)),
  (5, 302, 8, 1982, 'Il Trovatore', datetime.datetime(2008, 6, 5, 19, 0))])
```

```
# Use sql_head to get the head of sales
sql_head("sales")
```

|   | salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid | comm |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 36861 | 21191 | 7872 | 1875 | 4 | 728.00 |  |
| **1** | 2 | 4 | 8117 | 11498 | 4337 | 1983 | 2 | 76.00 |  |
| **2** | 3 | 5 | 1616 | 17433 | 8647 | 1983 | 2 | 350.00 |  |
| **3** | 4 | 5 | 1616 | 19715 | 8647 | 1983 | 1 | 175.00 |  |

## WHERE - 5 points

Now let's do a bit of filtering with WHERE. Write and run queries to get the following results. **LIMIT all returns to first five rows.**

- Get venues with >= 10000 seats from the venues table
- Get venues in Arizona
- Get users who have a first name that starts with H
- Get **just email addresses** of users who gave a .edu email address

```
# Get big venues... so those with >= than 10000 seats
sq = """select * from venue where venueseats >= 10000"""
run_query(sq)
```

```
    (['venueid', 'venuename', 'venuecity', 'venuestate', 'venueseats'],
     [(5, 'Gillette Stadium', 'Foxborough', 'MA', 68756),
      (6, 'New York Giants Stadium', 'East Rutherford', 'NJ', 80242),
      (15, 'McAfee Coliseum', 'Oakland', 'CA', 63026),
      (18, 'Madison Square Garden', 'New York City', 'NY', 20000),
      (67, 'Ralph Wilson Stadium', 'Orchard Park', 'NY', 73967),
      (68, 'Rogers Centre', 'Toronto', 'ON', 50516),
      (69, 'Dolphin Stadium', 'Miami Gardens', 'FL', 74916),
      (70, 'M&T Bank Stadium', 'Baltimore', 'MD', 70107),
      (71, 'Paul Brown Stadium', 'Cincinnati', 'OH', 65535),
      (72, 'Cleveland Browns Stadium', 'Cleveland', 'OH', 73200),
      (73, 'Heinz Field', 'Pittsburgh', 'PA', 65050),
      (74, 'Reliant Stadium', 'Houston', 'TX', 72000),
      (75, 'Lucas Oil Stadium', 'Indianapolis', 'IN', 63000),
      (76, 'Jacksonville Municipal Stadium', 'Jacksonville', 'FL', 73800),
      (77, 'LP Field', 'Nashville', 'TN', 68804),
      (78, 'INVESCO Field', 'Denver', 'CO', 76125),
      (79, 'Arrowhead Stadium', 'Kansas City', 'MO', 79451),
      (80, 'Qualcomm Stadium', 'San Diego', 'CA', 70561),
      (81, 'Texas Stadium', 'Irving', 'TX', 65595),
      (82, 'Lincoln Financial Field', 'Philadelphia', 'PA', 68532),
      (83, 'FedExField', 'Landover', 'MD', 91704),
      (84, 'Soldier Field', 'Chicago', 'IL', 63000),
      (85, 'Ford Field', 'Detroit', 'MI', 65000),
      (86, 'Lambeau Field', 'Green Bay', 'WI', 72922),
      (87, 'Hubert H. Humphrey Metrodome', 'Minneapolis', 'MN', 64035),
      (88, 'Georgia Dome', 'Atlanta', 'GA', 71149),
      (89, 'Bank of America Stadium', 'Charlotte', 'NC', 73298),
      (90, 'Louisiana Superdome', 'New Orleans', 'LA', 72000),
      (91, 'Raymond James Stadium', 'Tampa', 'FL', 65647),
      (93, 'Edward Jones Dome', 'St. Louis', 'MO', 66965),
      (94, 'Monster Park', 'San Francisco', 'CA', 69843),
      (95, 'Qwest Field', 'Seattle', 'WA', 67000),
      (96, 'Oriole Park at Camden Yards', 'Baltimore', 'MD', 48876),
      (97, 'Fenway Park', 'Boston', 'MA', 39928),
      (98, 'Yankee Stadium', 'New York City', 'NY', 52325),
      (99, 'Tropicana Field', 'St. Petersburg', 'FL', 36048),
      (100, 'U.S. Cellular Field', 'Chicago', 'IL', 40615),
      (101, 'Progressive Field', 'Cleveland', 'OH', 43345),
      (102, 'Comerica Park', 'Detroit', 'MI', 41782),
      (103, 'Kauffman Stadium', 'Kansas City', 'MO', 40793),
      (104, 'Angel Stadium of Anaheim', 'Anaheim', 'CA', 45050),
      (105, 'Safeco Field', 'Seattle', 'WA', 47116),
      (106, 'Rangers BallPark in Arlington', 'Arlington', 'TX', 49115),
      (107, 'Turner Field', 'Atlanta', 'GA', 50091),
      (109, 'Citizens Bank Park', 'Philadelphia', 'PA', 43647),
      (110, 'Nationals Park', 'Washington', 'DC', 41888),
      (111, 'Wrigley Field', 'Chicago', 'IL', 41118),
      (112, 'Great American Ball Park', 'Cincinnati', 'OH', 42059),
      (113, 'Minute Maid Park', 'Houston', 'TX', 40950),
      (114, 'Miller Park', 'Milwaukee', 'WI', 42200),
      (115, 'PNC Park', 'Pittsburgh', 'PA', 38496),
      (116, 'Busch Stadium', 'St. Louis', 'MO', 49660),
      (118, 'Coors Field', 'Denver', 'CO', 50445),
      (119, 'Dodger Stadium', 'Los Angeles', 'CA', 56000),
      (120, 'PETCO Park', 'San Diego', 'CA', 42445),
      (121, 'AT&T Park', 'San Francisco', 'CA', 41503),
      (126, 'Shoreline Amphitheatre', 'Mountain View', 'CA', 22000)])
```

```
# Get venues in AZ
sq = """select * from venue where venuestate = 'AZ'"""
run_query(sq)
```

```
    (['venueid', 'venuename', 'venuecity', 'venuestate', 'venueseats'],
     [(38, 'US Airways Center', 'Phoenix', 'AZ', 0),
      (65, 'Jobing.com Arena', 'Glendale', 'AZ', 0),
      (92, 'University of Phoenix Stadium', 'Glendale', 'AZ', 0),
      (117, 'Chase Field', 'Phoenix', 'AZ', 0)])
```

```
        (117, 'Chase Field', 'Phoenix', 'AZ', 0)])
```

```
# Get users who have a first name that starts with H
sq = """select * from users where firstname like 'H%'"""
run_query(sq)
```

```
     (['userid',
       'username',
       'firstname',
       'lastname',
       'city',
       'state',
       'email',
       'phone',
       'likesports',
       'liketheatre',
       'likeconcerts',
       'likejazz',
       'likeclassical',
       'likeopera',
       'likerock',
       'likevegas',
       'likebroadway',
       'likemusicals'],
      [(13,
        'QTF33MCG',
        'Henry',
        'Cochran',
        'Bossier City',
        'QC',
        'Aliquam.vulputate.ullamcorper@amalesuada.org',
        '(783) 105-0989',
        None,
        True,
        None,
        None,
        None,
        None,
        True,
        True,
        True,
        None),
       (22,
        'RHT62AGI',
        'Hermione',
        'Trevino',
        'Walnut',
        'WI',
        'non.justo.Proin@ametconsectetuer.edu',
        '(245) 110-6540',
        None,
        True,
        None,
        False,
        False
```

```
         None,
         None,
         True,
         False,
         None),
        (29,
         'HUH27PKK',
         'Helen',
         'Avery',
```

```
# Get all .edu email addresses... just the email addresses
sq = """select email from users where email like '%.edu'"""
run_query(sq)
```

```
     (['email'],
      [('Etiam.laoreet.libero@sodalesMaurisblandit.edu',),
       ('Suspendisse.tristique@nonnisiAenean.edu',),
       ('ullamcorper.nisl@Cras.edu',),
       ('vel.est@velitegestas.edu',),
       ('justo.nec.ante@quismassa.edu',),
       ('non.justo.Proin@ametconsectetuer.edu',),
       ('non@enimsitamet.edu',),
       ('cursus@pedeacurna.edu',),
       ('in.faucibus.orci@ultrices.edu',),
       ('magnis.dis.parturient@iaculisodioNam.edu',),
       ('sapien.molestie.orci@Vestibulumaccumsan.edu',),
       ('risus.Donec.egestas@Suspendissecommodo.edu',),
       ('mi.lorem@nunc.edu',),
       ('enim.gravida@placerataugue.edu',),
       ('in.lobortis.tellus@bibendumDonecfelis.edu',),
       ('quis@sitamet.edu',),
       ('ornare.facilisis.eget@sociisnatoquepenatibus.edu',),
       ('egestas.a.scelerisque@utnisi.edu',),
       ('dictum.augue.malesuada@arcueu.edu',),
       ('augue.scelerisque@pellentesque.edu',),
       ('sollicitudin.a@Nuncut.edu',),
       ('Vivamus.sit@fermentum.edu',),
       ('enim.gravida@atnisi.edu',),
       ('neque.tellus.imperdiet@Fuscealiquam.edu',),
       ('aliquam@euarcu.edu',),
       ('quam.Curabitur@veliteusem.edu',),
       ('nisi.sem.semper@tempor.edu',),
       ('arcu.Vestibulum@augue.edu',),
       ('ultrices.sit@Maecenaslibero.edu',),
       ('ante.blandit.viverra@variuset.edu',),
       ('Nulla@metusInlorem.edu',),
       ('ac.facilisis@tellus.edu',),
       ('in.faucibus.orci@velconvallisin.edu',),
       ('elementum.at@felisullamcorperviverra.edu',),
       ('Donec@luctus.edu',),
       ('quis.diam.luctus@amet.edu',),
       ('amet.risus.Donec@arcuiaculis.edu',),
       ('quis.urna@Phasellusfermentumconvallis.edu',),
       ('placerat.augue.Sed@tristique.edu',),
       ('Nunc.mauris@luctusipsum.edu',),
       ('Quisque@nondapibusrutrum.edu',),
       ('Nunc@quis.edu',),
       ('Cras@a.edu',),
       ('Proin.vel@tortor.edu',),
       ('dui.lectus@dictumsapienAenean.edu',),
       ('libero.Proin.sed@erat.edu',),
       ('nulla.Donec.non@estmauris.edu',),
       ('ultrices.mauris@egestasadui.edu',),
       ('eu.placerat.eget@maurisutmi.edu',),
       ('placerat@pharetrautpharetra.edu',),
       ('viverra.Donec@luctus.edu',),
       ('vulputate@pellentesquetellus.edu',),
       ('nisl.arcu@sodalesMauris.edu',),
       ('Nulla.interdum.Curabitur@euarcuMorbi.edu',),
       ('ac@velitAliquamnisl.edu',),
       ('Donec@egetmetuseu.edu',),
       ('posuere.cubilia.Curae;@acrisusMorbi.edu',),
```

## GROUP BY and HAVING - 5 points

Time to practice some GROUP BY and HAVING operations. Please write and run queries that do the following:

GROUP BY application

- Find the top five venues that hosted the most events: Alias the count of events as 'events_hosted'. Also return the venue ID
- Get the number of events hosted in each month. You'll need to use `date_part()` in your select to select just the months. Alias this as 'month' and then the count of the number of events hosted as 'events_hosted'.
- Get the top five sellers who made the most commission. Alias their total commission made as 'total_com'. Also get their average commission made and alias as 'avg_com'. Be sure to also display the seller_id.

HAVING application

- Using the same query as the last one, instead of getting the top five sellers get all sellers who have made a total commission greater than $4000.
- Using the same query as the first groupby, instead of returning the top five venues, return just the ID's of venues that have had greater than 60 events.

```
### GROUP BY application
# Find the top five venues that hosted the most events: Alias the count of events a
sql_head("event")
sq = """select venueid, count(venueid) as events_hosted from event group by venueic
run_query(sq)
```

```
    (['venueid', 'events_hosted'],
     [(220, 81), (217, 81), (203, 80), (222, 74), (216, 72)])
```

```
# Get the number of events hosted in each month. You'll need to use `date_part()` i
# Alias this as 'month' and then the count of the number of events hosted as 'event
sq = """select date_part('month', starttime) as month, count(eventid) as events_hos
run_query(sq)
```

```
    (['month', 'events_hosted'],
     [(1.0, 778),
      (2.0, 711),
      (3.0, 753),
      (4.0, 725),
      (5.0, 727),
      (6.0, 709),
      (7.0, 729),
      (8.0, 737),
      (9.0, 746),
      (10.0, 735),
      (11.0, 726),
      (12.0, 722)])
```

```
# Get the top five sellers who made the most commission. Alias their total commissi
# Also get their average commission made and alias as 'avg_com'. Be sure to also di
sq = """select sellerid, sum(commission) as total_com, avg(commission) as avg_com f
run_query(sq)
```

```
    (['sellerid', 'total_com', 'avg_com'],
     [(1140, Decimal('4859.85'), Decimal('347.1321428571428571')),
      (43551, Decimal('4704.75'), Decimal('470.4750000000000000')),
      (13385, Decimal('4274.25'), Decimal('388.5681818181818182')),
      (25433, Decimal('4147.95'), Decimal('518.4937500000000000')),
      (2372, Decimal('4073.85'), Decimal('678.9750000000000000'))])
```

```
### HAVING application
# Using the same query as the last groupby, instead of getting the top five sellers
sq = """select sellerid, sum(commission) as total_com, avg(commission) as avg_com f
run_query(sq)
```

```
    (['sellerid', 'total_com', 'avg_com'],
     [(1140, Decimal('4859.85'), Decimal('347.1321428571428571')),
      (43551, Decimal('4704.75'), Decimal('470.4750000000000000')),
      (13385, Decimal('4274.25'), Decimal('388.5681818181818182')),
      (25433, Decimal('4147.95'), Decimal('518.4937500000000000')),
      (2372, Decimal('4073.85'), Decimal('678.9750000000000000'))])
```

```
# Using the same query as the first groupby, instead of returning the top five venu
sq = """select venueid from event group by venueid having count(venueid) > 60"""
run_query(sq)
```

```
    (['venueid'],
     [(237,),
      (238,),
      (239,),
      (226,),
      (209,),
      (205,),
      (201,),
      (216,),
      (220,),
      (221,),
      (207,),
      (243,),
      (222,),
      (248,),
      (208,),
      (217,),
      (218,),
      (215,),
      (203,),
      (228,)])
```

## JOIN - 5 points

Time for some joins. You've probably noticed by now that there is at least one relational key in each table, but some have more. For example, sales has a unique sale id, listing id, seller id, buyer id, date id. This allows you to link each sale to relevant information in other tables.
**LIMIT each output to 5**

Please write queries to do the following items:

- Join information of users to each sale made.
- Join information about each venue to each event.

[ ] ↳ *6 cells hidden*

## Subqueries - 5 points

To wrap up let's do several subqueries. Please do the following:

- Get all purchases made by users of live in Arizona
- Get event information for all events that took place in a venue where the venue name ends with 'Stadium'.
- Get event information for all events where the total ticket sales were greater than $50,000.

```
# Get all purchases from users who live in Arizona
```

```
sq = """
select * from sales where sales.buyerid in (select userid from users where state ir
"""
run_query(sq)

     (['salesid',
       'listid',
       'sellerid',
       'buyerid',
       'eventid',
       'dateid',
       'qtysold',
       'pricepaid',
       'commission',
       'saletime'],
      [(43,
        47,
        49346,
        33489,
        8577,
        2141,
        2,
        Decimal('378.00'),
        Decimal('56.70'),
        datetime.datetime(2008, 11, 11, 9, 51, 6)),
       (79,
        101,
        37592,
        7079,
        3340,
        1878,
        1,
        Decimal('36.00'),
        Decimal('5.40'),
        datetime.datetime(2008, 2, 21, 10, 32, 10)),
       (81,
        103,
        26314,
        7079,
        15,
        2033,
        1,
        Decimal('181.00'),
        Decimal('27.15'),
        datetime.datetime(2008, 7, 26, 11, 4, 13)),
       (83,
        106,
        12538,
        7079,
        250,
        1884,
        1,
        Decimal('109.00'),
        Decimal('16.35'),
        datetime.datetime(2008, 2, 27, 11, 58, 35)),
  (154,
   162,
   27703,
   46451,
   2906,
   1907,
   3,
   Decimal('426.00'),
```

```
# Get event information for all events that took place in a venue where the name er
sq = """select * from event where event.venueid in (select venueid from venue where
run_query(sq)
```

```
([('eventid', 'venueid', 'catid', 'dateid', 'eventname', 'starttime'],
 [(3803,
   2,
   9,
   2181,
   'Dropkick Murphys',
   datetime.datetime(2008, 12, 21, 14, 0)),
  (3816, 11, 9, 2139, 'Keb Mo', datetime.datetime(2008, 11, 9, 19, 0)),
  (3821,
   79,
   9,
   1885,
   'Charlie Daniels Band',
   datetime.datetime(2008, 2, 28, 19, 30)),
  (3824, 98, 9, 1885, 'Govt Mule', datetime.datetime(2008, 2, 28, 14, 0)),
  (3835, 74, 9, 2073, 'LeAnn Rimes', datetime.datetime(2008, 9, 4, 14, 30)),
  (3837, 67, 9, 1850, 'Randy Travis', datetime.datetime(2008, 1, 24, 15, 0)),
  (3840, 14, 9, 1876, 'Gwen Stefani', datetime.datetime(2008, 2, 19, 19,
30)),
  (3843, 75, 9, 1986, 'Blake Shelton', datetime.datetime(2008, 6, 9, 14,
30)),
  (3854, 6, 9, 1982, 'Gwen Stefani', datetime.datetime(2008, 6, 5, 15, 0)),
  (3862, 116, 9, 1855, 'Teddy Geiger', datetime.datetime(2008, 1, 29, 19,
30)),
  (3864, 75, 9, 1992, 'Celine Dion', datetime.datetime(2008, 6, 15, 15, 0)),
  (3866, 72, 9, 1950, 'Cold War Kids', datetime.datetime(2008, 5, 1, 14, 0)),
  (3869, 67, 9, 2115, 'Dave Stewart', datetime.datetime(2008, 10, 16, 20,
0)),
  (3873, 11, 9, 2188, 'Avril Lavigne', datetime.datetime(2008, 12, 28, 14,
0)),
  (3874, 129, 9, 1885, 'America', datetime.datetime(2008, 2, 28, 15, 0)),
  (3889,
   67,
   9,
   2108,
   'Natasha Bedingfield',
   datetime.datetime(2008, 10, 9, 19, 30)),
  (3896,
   14,
   9,
   1893,
   'Natasha Bedingfield'
```

```
          'Natasha Bedingfield',
          datetime.datetime(2008, 3, 8, 15, 0)),
         (3899, 75, 9, 2089, 'Rusted Root', datetime.datetime(2008, 9, 20, 14, 0)),
         (3901, 70, 9, 1861, 'Squeeze', datetime.datetime(2008, 2, 4, 14, 0)),
         (3908,
          74,
          9,
          2098,
          'Sound Tribe Sector 9',
          datetime.datetime(2008, 9, 29, 14, 0)),
         (3909, 72, 9, 2190, 'Scorpions', datetime.datetime(2008, 12, 30, 14, 0)),
         (3912, 103, 9, 1926, 'Nofx', datetime.datetime(2008, 4, 10, 19, 0)),
         (3918, 74, 9, 1940, 'Gnarls Barkley', datetime.datetime(2008, 4, 24, 15,
       0)),
         (3925, 3, 9, 1864, 'Donny Osmond', datetime.datetime(2008, 2, 7, 14, 30)),
         (3926, 72, 9, 1848, 'Neil Diamond', datetime.datetime(2008, 1, 22, 19,
       30)),
```

```
# Get event information where the total sales for that event were greater than $50(
sq = """select * from event where eventid in (select eventid from sales group by ev
run_query(sq)
```

```
    (['eventid', 'venueid', 'catid', 'dateid', 'eventname', 'starttime'],
     [(289,
       300,
       8,
       2100,
       'Adriana Lecouvreur',
       datetime.datetime(2008, 10, 1, 19, 30)),
      (1602,
       257,
       6,
       2128,
       'Phantom of the Opera',
       datetime.datetime(2008, 10, 29, 19, 30)),
      (7895, 47, 9, 1944, 'Janet Jackson', datetime.datetime(2008, 4, 28, 15,
    0))])
```

```
get_table_names()
```

```
    [('users',),
     ('venue',),
     ('category',),
     ('date',),
     ('event',),
     ('listing',),
     ('sales',)]
```

```
get_column_names(table_name= 'date')
```

```
    [('dateid',),
     ('caldate',),
     ('day',),
     ('week',),
     ('month',),
     ('qtr',),
     ('year',),
     ('holiday',)]
```

```
sq = """ SELECT * FROM date
          LIMIT 5;"""
run_query(sq)
```

```
([['dateid', 'caldate', 'day', 'week', 'month', 'qtr', 'year', 'holiday'],
 [(1827, datetime.date(2008, 1, 1), 'WE ', 1, 'JAN  ', '1    ', 2008, True),
  (1828, datetime.date(2008, 1, 2), 'TH ', 1, 'JAN  ', '1    ', 2008, False),
  (1829, datetime.date(2008, 1, 3), 'FR ', 1, 'JAN  ', '1    ', 2008, False),
  (1830, datetime.date(2008, 1, 4), 'SA ', 2, 'JAN  ', '1    ', 2008, False),
  (1831, datetime.date(2008, 1, 5), 'SU ', 2, 'JAN  ', '1    ', 2008,
False)])
```