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!

```
[1] import psycopg2
    import pandas as pd
```

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

```
[2] # Make our connection/cursor function
    AWS_host_name = "ticketsdb.chwicm24ose3.us-east-1.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')
```

```
      port='5432')

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

    return(conn, cur)   # Return both the connection and the cursor

  # 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' """
```

```
[2]
    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
```

Q1 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 :)

```
## Q1 Your function starts here
# make sql_head function
def sql_head(table_name):
    conn, cur = get_conn_cur()
    query = f"SELECT * FROM {table_name} LIMIT 5"
    sql_head_df = pd.read_sql_query(query, conn)
    cur.close()
    conn.close()

    return sql_head_df

get_table_names()
```

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

```
[4] # Check that it works!
    sql_head(table_name = 'sales')
    ## Q1 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
<ipython-input-3-4e7c988bffb5>:6: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested
```

```
[4]
```

	sales_id	list_id	seller_id	buyer_id	event_id	date_id	qty_sold	price_paid	commission	sale_time
0	2	4	8117	11498	4337	1983	2	76	11.40	6/6/2008 05:00:16
1	3	5	1616	17433	8647	1983	2	350	52.50	6/6/2008 08:26:17
2	4	5	1616	19715	8647	1986	1	175	26.25	6/9/2008 08:38:52
3	5	6	47402	14115	8240	2069	2	154	23.10	8/31/2008 09:17:02
4	6	10	24858	24888	3375	2023	2	394	59.10	7/16/2008 11:59:24

Q2 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.

```
## Q2 Your function starts here
# Getting table names
get_table_names()
```

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

```
[6] # Getting column names
    for i in get_table_names():
      print(get_column_names(table_name = i[0]))
```

```
[('sales_id',), ('list_id',), ('seller_id',), ('buyer_id',), ('event_id',), ('date_id',), ('qty_sold',), ('price_paid',), ('commission',), ('sale_time',)]
[('venue_id',), ('venue_seats',), ('venue_name',), ('venue_city',), ('venue_state',)]
[('event_id',), ('venue_id',), ('cat_id',), ('date_id',), ('start_time',), ('event_name',)]
[('user_id',), ('user_name',), ('first_name',), ('last_name',), ('city',), ('state',), ('email',), ('phone',), ('like_sports',), ('like_theatre',), ('like_concerts',), ('like_jazz',), ('like_classica
[('cat_id',), ('cat_group',), ('cat_name',), ('cat_desc',)]
[('list_id',), ('seller_id',), ('event_id',), ('date_id',), ('num_tickets',), ('price_per_ticket',), ('total_price',), ('list_time',)]
```

```
[('date_id',), ('qtr',), ('year',), ('holiday',), ('week',), ('cal_date',), ('day',), ('month',)]
```

```
# 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]
```

```
[[('sales_id',),
  ('list_id',),
  ('seller_id',),
  ('buyer_id',),
  ('event_id',),
  ('date_id',),
  ('qty_sold',),
  ('price_paid',),
  ('commission',),
  ('sale_time',)],
 [('venue_id',),
  ('venue_seats',),
  ('venue_name',),
  ('venue_city',),
  ('venue_state',)],
 [('event_id',),
  ('venue_id',),
  ('cat_id',),
  ('date_id',),
  ('start_time',),
  ('event_name',)],
 [('user_id',),
  ('user_name',),
  ('first_name',),
  ('last_name',),
  ('city',),
  ('state',),
  ('email',),
  ('phone',),
  ('like_sports',),
  ('like_theatre',),
  ('like_concerts',),
  ('like_jazz',),
  ('like_classical',),
  ('like_opera',),
  ('like_rock',),
  ('like_vegas',),
  ('like_broadway',),
  ('like_musicals',)],
 [('cat_id',), ('cat_group',), ('cat_name',), ('cat_desc',)],
 [('list_id',),
  ('seller_id',),
  ('event_id',),
```

```
# Query on events
sq = """ SELECT *
FROM event
LIMIT 5"""
run_query(sq)
```

```
(['event_id', 'venue_id', 'cat_id', 'date_id', 'event_name', 'start_time'],
 [(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)),
  (6,
   308,
   8,
   2109,
   'L Elisir d Amore',
   datetime.datetime(2008, 10, 10, 19, 30))])
```

```
[9] # Use sql_head to get the head of sales
sql_head(table_name = 'sales')
## Q2 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
<ipython-input-3-4e7c988bffb5>:6: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested
  sql_head_df = pd.read_sql_query(query, conn)
```

	sales_id	list_id	seller_id	buyer_id	event_id	date_id	qty_sold	price_paid	commission	sale_time
0	2	4	8117	11498	4337	1983	2	76	11.40	6/6/2008 05:00:16
1	3	5	1616	17433	8647	1983	2	350	52.50	6/6/2008 08:26:17
2	4	5	1616	19715	8647	1986	1	175	26.25	6/9/2008 08:38:52
3	5	6	47402	14115	8240	2069	2	154	23.10	8/31/2008 09:17:02
4	6	10	24858	24888	3375	2023	2	394	59.10	7/16/2008 11:59:24

Q3 WHERE - 5 points

Now let's do a bit of filtering with WHERE. Write and run queries to get the following results.

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 (**full row of data, including venue_id, etc.**) with >= 10000 seats from the venues table
- Get venues (**full row of data, including venue_id, etc.**) in Arizona
- Get users (**just the first names**) who have a first name that starts with H
- Get the email addresses (**just the email addresses**) of users who gave a .edu email address

```
## Q3 Your function starts here
# Get big venues... so those with >= than 10000 seats
sq1 = """ SELECT *
          FROM venue
          WHERE venue_seats >= 10000
          LIMIT 5"""

run_query(sq1)
```

```
(['venue_id', 'venue_name', 'venue_city', 'venue_state', 'venue_seats'],
 [(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)])
```

```
[11] # Get venues in AZ
     sq2 = """ SELECT *
               FROM venue
               WHERE venue_state = 'AZ'
               LIMIT 5;"""
     run_query(sq2)
```

```
(['venue_id', 'venue_name', 'venue_city', 'venue_state', 'venue_seats'],
 [(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)])
```

```
[12] #Get users who have a first name that starts with H
     sq3 = """ SELECT first_name
               FROM users
               WHERE first_name LIKE 'H%'
               LIMIT 5;"""
     run_query(sq3)
```

```
(['first_name'],
 [('Henry',), ('Hermione',), ('Helen',), ('Howard',), ('Herman',)])
```

```
[13] # Get all .edu email addresses... just the email addresses
     sq4 = """ SELECT email
               FROM users
               WHERE email LIKE '%.edu'
               LIMIT 5;"""
     run_query(sq4)
     ## Q3 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
(['email'],
 [('Suspendisse.tristique@nonnisiAenean.edu',),
  ('ullamcorper.nisl@Cras.edu',),
  ('vel.est@velitegestas.edu',),
  ('justo.nec.ante@quismassa.edu',),
  ('non.justo.Proin@ametconsectetuer.edu',)])
```

Q4 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.

```
[14] ## Q4 Your function starts here
     ### 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
     sq1 = """ SELECT venue_id,
               COUNT(*) AS events_hosted
```

```
[14]            FROM venue
                GROUP BY venue_id
                ORDER BY events_hosted DESC
                LIMIT 5;"""
    run_query(sq1)
```

```
(['venue_id', 'events_hosted'],
 [(87, 1), (71, 1), (229, 1), (68, 1), (116, 1)])
```

```
    # 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'
    sq2 = """ SELECT date_part('month', start_time) AS month,
              COUNT(*) AS events_hosted
              FROM event
              GROUP BY month
              ORDER BY month;"""
    run_query(sq2)
```

```
(['month', 'events_hosted'],
 [(1.0, 777),
  (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)])
```

```
[16] sql_head(table_name = 'event')
```

```
<ipython-input-3-4e7c988bffb5>:6: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested
  sql_head_df = pd.read_sql_query(query, conn)
```

	event_id	venue_id	cat_id	date_id	event_name	start_time
0	2	306	8	2114	Boris Godunov	2008-10-15 20:00:00
1	3	302	8	1935	Salome	2008-04-19 14:30:00
2	4	309	8	2090	La Cenerentola (Cinderella)	2008-09-21 14:30:00
3	5	302	8	1982	Il Trovatore	2008-06-05 19:00:00
4	6	308	8	2109	L Elisir d Amore	2008-10-10 19:30:00

```
    # 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
    sq3 = """ SELECT seller_id,
           SUM(commission) AS total_com,
           AVG(commission) AS avg_com
           FROM sales
           GROUP BY seller_id
           ORDER BY total_com DESC
           LIMIT 5;"""
    run_query(sq3)
```

```
(['seller_id', 'total_com', 'avg_com'],
 [(1140, 4859.85, 347.13214285714287),
  (43551, 4704.75, 470.475),
  (13385, 4274.25, 388.5681818181818),
  (25433, 4147.95, 518.49375),
  (2372, 4073.85, 678.975)])
```

```
[18] ### HAVING application
    # Using the same query as the last groupby, instead of getting the top five sellers get all sellers who have made a total commission greater than $4000
    sq4 = """ SELECT seller_id,
              SUM(commission) AS total_com,
              AVG(commission) AS avg_com
              FROM sales
              GROUP BY seller_id
              HAVING SUM(commission) > 4000
              ORDER BY total_com DESC;"""
    run_query(sq4)
```

```
(['seller_id', 'total_com', 'avg_com'],
 [(1140, 4859.85, 347.13214285714287),
  (43551, 4704.75, 470.475),
  (13385, 4274.25, 388.5681818181818),
  (25433, 4147.95, 518.49375),
  (2372, 4073.85, 678.975)])
```

```
[19] # 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
    sq5 = """ SELECT venue_id
              FROM event
              GROUP BY venue_id
              HAVING COUNT(*) > 60
              LIMIT 5;"""
    run_query(sq5)
    ## Q4 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
(['venue_id'], [(237,), (238,), (239,), (226,), (209,)])
```

Q5 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.

```
## Q5 Your function starts here
sq1 = """ SELECT * FROM sales
          LIMIT 5;"""
run_query(sq1)
```

```
(['sales_id',
  'list_id',
  'seller_id',
  'buyer_id',
  'event_id',
  'date_id',
  'qty_sold',
  'price_paid',
  'commission',
  'sale_time'],
 [(2, 4, 8117, 11498, 4337, 1983, 2, 76, 11.4, '6/6/2008 05:00:16'),
  (3, 5, 1616, 17433, 8647, 1983, 2, 350, 52.5, '6/6/2008 08:26:17'),
  (4, 5, 1616, 19715, 8647, 1986, 1, 175, 26.25, '6/9/2008 08:38:52'),
  (5, 6, 47402, 14115, 8240, 2069, 2, 154, 23.1, '8/31/2008 09:17:02'),
  (6, 10, 24858, 24888, 3375, 2023, 2, 394, 59.1, '7/16/2008 11:59:24')])
```

```
[21] sq2 = """ SELECT * FROM users
               LIMIT 5;"""
     run_query(sq2)
```

```
   'True',
   None,
   None,
   'True'),
  (4,
   'XDZ38RDD',
   'Barry',
```

```
[22] get_column_names(table_name='users')
```

```
[('user_id',),
 ('user_name',),
 ('first_name',),
 ('last_name',),
 ('city',),
 ('state',),
 ('email',),
 ('phone',),
 ('like_sports',),
 ('like_theatre',),
 ('like_concerts',),
 ('like_jazz',),
 ('like_classical',),
 ('like_opera',),
 ('like_rock',),
 ('like_vegas',),
 ('like_broadway',),
 ('like_musicals',)]
```

```
[23] get_column_names(table_name='sales')
```

```
[('sales_id',),
 ('list_id',),
 ('seller_id',),
 ('buyer_id',),
 ('event_id',),
 ('date_id',),
 ('qty_sold',),
 ('price_paid',),
 ('commission',),
 ('sale_time',)]
```

```
# Join users information to each sale
sq3 = """ SELECT *
          FROM sales
          JOIN users
          ON sales.buyer_id = users.user_id
          LIMIT 5;"""
run_query(sq3)
```

```
(['sales_id',
  'list_id',
  'seller_id',
  'buyer_id',
  'event_id',
```

```
    'last_name',
    'city',
    'state',
    'email',
    'phone',
    'like_sports',
    'like_theatre',
    'like_concerts',
    'like_jazz',
    'like_classical',
    'like_opera',
    'like_rock',
    'like_vegas',
    'like_broadway',
    'like_musicals'],
   [(4,
     5,
     1616,
     19715,
     8647,
     1986,
     1,
     175,
     26.25,
     '6/9/2008 08:38:52',
     19715,
     'PZY80KEH',
     'Alexander',
     'Spencer',
     'Jordan Valley',
     'ID',
     'amet@est.edu',
     '(963) 349-0806',
     'False',
     'False',
     None,
     'False',
     None,
     None,
     None,
     None,
     'False',
     None),
    (5,
     6,
     47402,
     14115,
     8240,
```

```
# For each event attach the venue information
sq4 = """ SELECT *
          FROM event
          JOIN venue
          ON event.venue_id = venue.venue_id
          LIMIT 5;"""
run_query(sq4)
## Q5 Your function ends here - Any code outside of these start/end markers won't be graded
```

```
    'venue_city',
    'venue_state',
    'venue_seats'],
   [(2,
     306,
     8,
     2114,
     'Boris Godunov',
     datetime.datetime(2008, 10, 15, 20, 0),
     306,
     'Lyric Opera House',
     'Baltimore',
     'MD',
     0),
    (3,
     302,
     8,
     1935,
     'Salome',
     datetime.datetime(2008, 4, 19, 14, 30),
     302,
     'Detroit Opera House',
     'Detroit',
     'MI',
     0),
    (4,
     309,
     8,
     2090,
     'La Cenerentola (Cinderella)',
     datetime.datetime(2008, 9, 21, 14, 30),
     309,
     'Los Angeles Opera',
     'Los Angeles',
     'CA',
     0),
    (5,
     302,
     8,
     1982,
```

Q 6 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.

```
## Q6 Your function starts here
# Get all purchases from users who live in Arizona
sq1 = """ SELECT *
          FROM sales
          WHERE buyer_id IN (SELECT user_id
          FROM users
          WHERE state = 'AZ');"""
run_query(sq1)
```

```
    4261,
    2041,
    3,
    711,
    106.65,
    '8/3/2008 04:20:11'),
   (93241,
    106446,
    45502,
    28585,
    8323,
    1997,
    1,
    274,
    41.1,
    '6/20/2008 06:10:01'),
   (93246,
    106454,
    29720,
    28347,
    1380,
    2130,
    1,
    442,
    66.3,
    '10/31/2008 06:17:07'),
   (93282,
    106503,
    39192,
```

```
# Get event information for all events that took place in a venue where the name ended in 'Stadium'
sq2 = """ SELECT e.*
          FROM event e
          JOIN venue v ON e.venue_id = v.venue_id
          WHERE v.venue_name LIKE '%Stadium';"""
run_query(sq2)
```

```
    89,
    9,
    2182,
    'Flogging Molly',
    datetime.datetime(2008, 12, 22, 19, 30)),
   (8507, 119, 9, 1995, 'Linda Eder', datetime.datetime(2008, 6, 18, 19, 0)),
   (8522, 11, 9, 2108, 'Nas', datetime.datetime(2008, 10, 9, 19, 0)),
   (8528, 79, 9, 1845, 'America', datetime.datetime(2008, 1, 19, 19, 30)),
   (8529,
    119,
    9,
    2135,
    'Vanessa Hudgens',
    datetime.datetime(2008, 11, 5, 19, 30)),
   (8549, 92, 9, 1954, 'Marc Anthony', datetime.datetime(2008, 5, 1, 19, 30)),
   (8553, 92, 9, 1883, 'John Prine', datetime.datetime(2008, 2, 26, 15, 0)),
   (8564, 80, 9, 2002, 'Phil Vassar', datetime.datetime(2008, 6, 25, 19, 0)),
   (8569, 74, 9, 1989, 'Girl Talk', datetime.datetime(2008, 6, 12, 14, 0)),
   (8573, 13, 9, 1898, 'Journey', datetime.datetime(2008, 3, 13, 14, 0)),
   (8577,
    70,
    9,
    2155,
    'Country Thunder USA',
    datetime.datetime(2008, 11, 25, 15, 0)),
   (8585,
    69,
    9,
    2028,
    'Little River Band',
    datetime.datetime(2008, 7, 21, 15, 0)),
   (8598, 70, 9, 2043, 'Bow Wow', datetime.datetime(2008, 8, 5, 15, 0)),
   (8599, 74, 9, 2174, 'John Hiatt', datetime.datetime(2008, 12, 14, 14, 0)),
   (8601, 75, 9, 2132, 'Built To Spill', datetime.datetime(2008, 11, 2, 14, 0)),
   (8602,
    3,
    9,
    2031,
    'Toad The Wet Sprocket',
    datetime.datetime(2008, 7, 24, 15, 0)),
   (8610, 71, 9, 1961, 'Sugarland', datetime.datetime(2008, 5, 1, 19, 0)),
   (8620,
```

```
[28] # Get event information where the total sales for that event were greater than $50000
     sq3 = """ SELECT e.*
               FROM event e
               WHERE e.event_id IN (
               SELECT s.event_id
               FROM sales s
               GROUP BY s.event_id
               HAVING SUM(s.price_paid) > 50000);"""
     run_query(sq3)
```

```
(['event_id', 'venue_id', 'cat_id', 'date_id', 'event_name', 'start_time'],
 [(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()
```

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

```
[30] get_column_names(table_name= 'date')
```

```
[('date_id',),
 ('qtr',),
 ('year',),
 ('holiday',),
 ('week',),
 ('cal_date',),
 ('day',),
 ('month',)]
```

```
## Q6 Your function ends here - Any code outside of these start/end markers won't be graded
```