In [2]:

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

In [3]:

```
conn = psycopg2.connect(host="localhost", port = 5432, database="MobileDevices", user="postgre
s", password="calpoly")
cur = conn.cursor()
```

In [4]:

```
def out():
    query_results = pd.DataFrame(cur.fetchall())
    return query_results
```

In [4]:

```
cur.execute(
    """select *
    from devices""")
out()
```

Out[4]:

	0	1	2	3	4	5	6	
0	2	Audiovox	CDM-8300	PP4-TX30B	Candybar	117.000000	41.000000	15.00000
1	3	Audiovox	CDM-9100	CJ6DCE42903A	Candybar	127.000000	46.000000	20.00000
2	4	Audiovox	CDM-9150x	CJ6DCE44941A	Candybar	127.000000	46.000000	20.00000
3	5	Audiovox	SMT5600	NM8TP	Candybar	108.000000	46.000000	16.00000
4	6	Audiovox	VI-600	PP4TX-60B	Candybar	109.000000	46.000000	18.00000
...	...	...	...	...	...	...	...	
2452	3379	Motorola	Motofone F3 Red	None	Candybar	114.000000	47.000000	9.00000
2453	3380	Motorola	W220 Pink	None	Clamshell	95.000000	46.700001	16.70000
2454	3381	Nokia	5300 XpressMusic Grey	PPIRM-146	Slider	92.000000	48.000000	21.00000
2455	3382	Samsung	SGH-X180 Gold	None	Clamshell	86.199997	42.000000	19.00000
2456	3383	Samsung	SGH-X180 Pink	None	Clamshell	86.199997	42.000000	19.00000

2457 rows × 13 columns

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

(1) Produce a list of devices with manufacturer “Motorola”?

In [78]:

```
cur.execute(
    """
    select *
    from devices d
    where manufacturer = 'Motorola'
    """)
out()
```

Out[78]:

	0	1	2	3	4	5	6	7	8
0	52	Motorola	V120t	IHDT56CA1	Candybar	127.000000	43.000000	28.000000	128.
1	53	Motorola	A388	IHDT6BK1	PDA	98.000000	58.000000	23.000000	130.
2	54	Motorola	A845	IHDT56EJ1	Candybar	101.000000	48.000000	20.000000	94.
3	55	Motorola	C331T	IHDT56CF1	Candybar	107.000000	48.000000	23.000000	99.
4	56	Motorola	C331	IHDT56CE1	Candybar	107.000000	46.000000	22.000000	80.
...	...	...	...	...	...	...	...	...	..
367	3339	Motorola	RAZR MAXX VE	IHDT56GJ2	Clamshell	101.000000	53.000000	15.000000	110.
368	3343	Motorola	RIZR Z3 Rose	IHDT56GY1	Slider	105.500000	45.500000	16.000000	115.
369	3350	Motorola	RAZR V3xx Platinum	None	Clamshell	86.300003	47.000000	24.400000	96.
370	3379	Motorola	Motofone F3 Red	None	Candybar	114.000000	47.000000	9.000000	70.
371	3380	Motorola	W220 Pink	None	Clamshell	95.000000	46.700001	16.700001	93.

372 rows × 13 columns

(2) Produce a list of devices with manufacturer “Motorola”? and having start_year = “2005”!

In [79]:

```
cur.execute(
    """
    select manufacturer, model_name, start_year
    from devices
    where manufacturer = 'Motorola'
    and start_year = 2005
    limit 5
    """)
out()
```

Out[79]:

	0	1	2
0	Motorola	i275	2005
1	Motorola	i850	2005
2	Motorola	E815	2005
3	Motorola	i760	2005
4	Motorola	i560 Black	2005

(3) Make a query that displays only one row of information: a count of the total number of devices in the database.

In [80]:

```
cur.execute(
    """
    select count(device_id) as total_num
    from devices
    """)
out()
```

Out[80]:

	0
0	2457

(4) Make a query, displaying only one row, that shows the number of devices for ‘Motorola” with start_year = “2006”

In [81]:

```
cur.execute(
    """
    select count(device_id) as total_num
    from devices
    where manufacturer = 'Motorola' and start_year = 2006
    """)
out()
```

Out[81]:

	0
0	35

(5) Make a query that displays only a single row for each manufacturer name and showing the number of devices with start_year = “2006” for each

In [82]:

```
cur.execute(
    """
select count(device_id) as num_of_devices, manufacturer
from devices
where start_year = 2006
group by manufacturer
order by manufacturer
limit 5
    """)
out()
```

Out[82]:

	0	1
0	8	Alcatel
1	1	BenQ-Siemens
2	4	Cingular
3	3	Danger
4	5	HP

(6) How heavy, on average, are devices with has_full_keyboard = ‘Yes’ ? Display the query and the answer below.

In [83]:

```
cur.execute("""select has_full_keyboard, avg(dim_weight) as avg_weight
from devices
where has_full_keyboard = 'Yes'
group by has_full_keyboard
order by avg_weight""")
out()
```

Out[83]:

```
     0           1
0  Yes  156.179487
```

(7) Make a two-column query that the year and the average device weight. Make this query have only one row for each year.

In [84]:

```
cur.execute("""
select start_year, avg(dim_weight) as avg_weight
from devices
group by start_year
order by start_year""")
out()
```

Out[84]:

```
        0           1
0  2000.0  139.280000
1  2001.0  128.522388
2  2002.0  117.203540
3  2003.0  112.639241
4  2004.0  112.046392
5  2005.0  110.349398
6  2006.0  141.170040
7  2007.0   95.358209
8     NaN  106.258540
```

(8) Make a query that shows the most common “form factor”, ie the shape of the phone, in the year 2007

In [85]:

```
cur.execute("""
select count(device_id) as qty, form_factor
from devices
where start_year = 2007
group by form_factor
order by qty desc
""")
out()
```

Out[85]:

```
    0             1
0  64     Clamshell
1  35     Candybar
2  29        Slider
3   5           PDA
4   2  Swivel/pivot
5   2       Special
```

(9) Calculate the average device’s density as a function of the starting year. This query will make use of fields “dim_height”, “dim_width”, “dim_depth”, and “dim_weight”. Density is defined as weight / volume. Are devices becoming more dense as time goes on? Show your query below, and the answer.

In [86]:

```
cur.execute("""
select avg(dim_weight/(dim_height*dim_width*dim_depth)) as volume, start_year
from devices
where dim_height != 0
and dim_width != 0
and dim_depth != 0
and dim_weight != 0
and dim_weight != 0
and start_year is not null
group by start_year
order by start_year
""")
out()
```

Out[86]:

	0	1
0	0.001124	2000
1	0.001263	2001
2	0.000978	2002
3	0.000967	2003
4	0.001089	2004
5	0.001115	2005
6	0.001482	2006
7	0.001187	2007

In [87]:

```
conn = psycopg2.connect(host="localhost", port = 5432, database="IMDB", user="postgres", password="calpoly")
cur = conn.cursor()
```

In [88]:

```
cur.execute("""
select *
from imdb_movie""")
out()
```

Out[88]:

	0	1	2	3	4	5	6	7	8	9
0	4	Django Unchained	With the help of a German bounty hunter, a fre...	2012	8.5	704215.0	165	R	1.0	/title/tt1853728/
1	5	A Million Ways to Die in the West	As a cowardly farmer begins to fall for the my...	2014	6.2	91380.0	116	R	2.0	/title/tt2557490/
2	6	The Hateful Eight	In post-Civil War Wyoming, bounty hunters try ...	2015	0.0	NaN	0	None	1.0	/title/tt3460252/
3	7	The Good, the Bad and the Ugly	A bounty hunting scam joins two men in an unea...	1966	8.9	410583.0	161	TV_14	3.0	/title/tt0060196/
4	8	The Salvation	In 1870s America, a peaceful American settler ...	2014	6.8	8303.0	92	None	4.0	/title/tt2720680/
...	...	...	...	...	...	...	...	...	...	...
4238	4273	Fantastic Beasts and Where to Find Them	None	2016	0.0	NaN	0	None	NaN	/title/tt3183660/
4239	4274	Bedknobs and Broomsticks	An apprentice witch, three kids and a cynical ...	1971	7.0	22332.0	117	G	50.0	/title/tt0066817/
4240	4275	Popeye	The adventures of the sailor man and his frien...	1980	5.1	21149.0	114	PG	142.0	/title/tt0081353/
4241	4276	Parental Guidance	Artie and Diane agree to look after their thre...	2012	6.0	18932.0	105	PG	694.0	/title/tt1047540/
4242	4277	Conan the Destroyer	Conan leads a ragtag group of adventurers on a...	1984	5.8	53962.0	103	PG	330.0	/title/tt0087078/

4243 rows × 14 columns

Show me a list of all movies directed by Martin Scorsese

In [89]:

```
cur.execute("""
select title, director_name, m.director_id
from imdb_movie m
join imdb_director d on d.director_id = m.director_id
where director_name = 'Martin Scorsese'
limit 5
""")
out()
```

Out[89]:

	0	1	2
0	Kundun	Martin Scorsese	374
1	The Wolf of Wall Street	Martin Scorsese	374
2	Hugo	Martin Scorsese	374
3	The Departed	Martin Scorsese	374
4	Shutter Island	Martin Scorsese	374

Show me a list of all movies starring Reese Witherspoon

In [90]:

```
cur.execute("""
select title, actor_name
from imdb_actor a
join imdb_actor_movie_map am on am.actor_id = a.actor_id
join imdb_movie m on m.movie_id = am.movie_id
where actor_name = 'Reese Witherspoon'
limit 5
""")
out()
```

Out[90]:

	0	1
0	This Means War	Reese Witherspoon
1	Election	Reese Witherspoon
2	Legally Blonde	Reese Witherspoon
3	Walk the Line	Reese Witherspoon
4	Cruel Intentions	Reese Witherspoon

List the movie titles directed by "Steven Spielberg".

In [91]:

```
cur.execute("""
select title, director_name
from imdb_movie m
join imdb_director d on d.director_id = m.director_id
where director_name = 'Steven Spielberg'
limit 5
""")
out()
```

Out[91]:

	0	1
0	Saving Private Ryan	Steven Spielberg
1	Lincoln	Steven Spielberg
2	Empire of the Sun	Steven Spielberg
3	War Horse	Steven Spielberg
4	1941	Steven Spielberg

How many movies are directed by "Steven Spielberg"? Show only one row

In [92]:

```
cur.execute("""
select count(title)
from imdb_movie m
join imdb_director d on d.director_id = m.director_id
where director_name = 'Steven Spielberg'
""")
out()
```

Out[92]:

	0
0	27

How many movies are in the genre "Western"?

In [93]:

```
cur.execute("""
select count(m.movie_id), genre_name
from imdb_genre g
join imdb_genre_movie_map gm on gm.genre_id = g.genre_id
join imdb_movie m on m.movie_id = gm.movie_id
where genre_name = 'Western'
group by genre_name
""")
out()
```

Out[93]:

	0	1
0	285	Western

List movies directed by "Steven Spielberg" that also are of genre "Action" and were released in the 1980's

In [94]:

```
cur.execute("""
select *
from imdb_movie m
join imdb_director d on d.director_id = m.director_id
join imdb_genre_movie_map gm on gm.movie_id = m.movie_id
join imdb_genre g on g.genre_id = gm.genre_id
where director_name = 'Steven Spielberg' and genre_name = 'Action'
and year_released between 1980 and 1989
""")
out()
```

Out[94]:

	0	1	2	3	4	5	6	7	8	9	
0	1292	Raiders of the Lost Ark	Archaeologist and adventurer Indiana Jones is ...	1981	8.6	535043	115	PG	207	/title/tt0082971/	1800000
1	1332	Indiana Jones and the Last Crusade	When Dr. Henry Jones Sr. suddenly goes missing...	1989	8.3	412544	127	None	207	/title/tt0097576/	4800000
2	1345	Indiana Jones and the Temple of Doom	After arriving in India, Indiana Jones is aske...	1984	7.6	260850	118	PG	207	/title/tt0087469/	2800000

Which genre is the most profitable on average? Here we define profit as (gross_usa – budget)

In [95]:

```
cur.execute("""
select genre_name, avg(gross_usa - budget) as avg_profit
from imdb_genre g
join imdb_genre_movie_map gm on g.genre_id = gm.genre_id
join imdb_movie m on m.movie_id = gm.movie_id
where gross_usa is not null and budget is not null
group by genre_name
order by avg_profit desc
limit 5
""")
out()
```

Out[95]:

	0	1
0	Animation	3.515049e+07
1	Family	3.060416e+07
2	Fantasy	2.688735e+07
3	Comedy	2.666887e+07
4	Adventure	2.576949e+07

Which genre has the highest average ROI. Here we define ROI as being

ROI = (gross_usa – budget) / (budget). Show only the top result

In [96]:

```
cur.execute("""
select genre_name, avg((gross_usa - budget)/budget) as avg_ROI
from imdb_genre g
join imdb_genre_movie_map gm on g.genre_id = gm.genre_id
join imdb_movie m on m.movie_id = gm.movie_id
where gross_usa is not null and budget is not null
group by genre_name
order by avg_ROI desc
limit 5
""")
out()
```

Out[96]:

	0	1
0	Horror	43.864908
1	Mystery	11.369758
2	Musical	2.840303
3	Sport	2.679475
4	Music	2.333051

Which director has generated the most overall profit (sum)? We define profit for a single movie as being P = (gross_usa – budget)

In [97]:

```
cur.execute('''
select director_name, sum(gross_usa - budget) as total_prof
from imdb_director d
join imdb_movie m on d.director_id = m.director_id
where gross_usa is not null and budget is not null
group by director_name
order by total_prof desc
''')
out()
```

Out[97]:

```
                     0              1
   0    Steven Spielberg   1.572858e+09
   1       James Cameron   1.185643e+09
   2     Chris Columbus    9.551148e+08
   3        George Lucas   8.635132e+08
   4       Peter Jackson   8.143276e+08
 ...                 ...            ...
1104         Yimou Zhang  -1.324253e+08
1105         Carl Rinsch  -1.367027e+08
1106  Paul W.S. Anderson  -1.668358e+08
1107      Andrew Stanton  -1.996898e+08
1108         Walter Hill  -2.062656e+08

1109 rows × 2 columns
```

What’s the average ROI of movies containing the actor: “Leonardo DiCaprio”

What’s the average ROI of movies containing the actor: “Nicolas Cage”

Make one query to answer both questions.

In [98]:

```
cur.execute('''
select actor_name, avg((gross_usa - budget)/budget) as ROI
from imdb_movie m
join imdb_actor_movie_map am on am.movie_id = m.movie_id
join imdb_actor a on a.actor_id = am.actor_id
where gross_usa is not null
and budget is not null
and (actor_name = 'Leonardo DiCaprio' or actor_name = 'Nicolas Cage')
group by actor_name
''')
out()
```

Out[98]:

```
                   0          1
0  Leonardo DiCaprio   0.476178
1       Nicolas Cage   0.251316
```

Identify the most profitable movie of all time (in this dataset)? Who is the director?

Profit = (gross_usa – budget). Show only the top result.

In [99]:

```
cur.execute('''
select title, director_name, (gross_usa - budget) as prof
from imdb_movie m
join imdb_director d on d.director_id = m.director_id
where gross_usa is not null
and budget is not null
order by prof desc
limit 1
''')
out()
```

Out[99]:

```
        0              1              2
0  Avatar  James Cameron    523507625.0
```

In [100]:

```
conn = psycopg2.connect(host="localhost", port = 5432, database="Northwind", user="postgres",
password="calpoly")
cur = conn.cursor()
```

Produce a query that lists Discontinued Products.

In [101]:

```
cur.execute('''
select *
from n_products
where discontinued = true
limit 5
''')
out()
```

Out[101]:

	0	1	2	3	4	5	6	7	8	9
0	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	True
1	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	True
2	17	Alice Mutton	7	6	20 - 1 kg tins	39.00	0	0	0	True
3	24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.50	20	0	0	True
4	28	Rössle Sauerkraut	12	7	25 - 825 g cans	45.60	26	0	0	True

Produce a query that lists a Count of Suppliers by Country. Produce a list in descending order.

In [102]:

```
cur.execute('''
select country, count(supplierID) as qty
from n_suppliers
group by country
order by qty desc
limit 5
''')
out()
```

Out[102]:

	0	1
0	USA	4
1	Germany	3
2	France	3
3	Sweden	2
4	Italy	2

Figure out a way to show total Revenue generated by each employee. Revenue is here defined as: R = (UnitPrice *Quantity)* (1 - Discount)

In [103]:

```
cur.execute('''
select e.employeeid, firstname, lastname, sum(unitprice*quantity * (1 - discount)) as total_revenue
from n_orders o
join n_employees e on e.employeeid = o.employeeid
join n_order_details od on od.orderid = o.orderid
group by e.employeeid, firstname, lastname
''')
out()
```

Out[103]:

	0	1	2	3
0	4	Margaret	Peacock	232890.845736
1	7	Robert	King	124568.234814
2	9	Anne	Dodsworth	77308.066403
3	6	Michael	Suyama	73913.129433
4	3	Janet	Leverling	202812.842851
5	1	Nancy	Davolio	192107.604371
6	5	Steven	Buchanan	68792.282437
7	2	Andrew	Fuller	166537.754835
8	8	Laura	Callahan	126862.277406

What are the most sold products? Show a list in descending order.

In [104]:

```
cur.execute('''
select sum(quantity) as total_sold, p.productid, productname
from n_products p
join n_order_details od on od.productid = p.productid
group by p.productid, productname
order by total_sold desc
''')
out()
```

Out[104]:

	0	1	2
0	1577	60	Camembert Pierrot
1	1496	59	Raclette Courdavault
2	1397	31	Gorgonzola Telino
3	1263	56	Gnocchi di nonna Alice
4	1158	16	Pavlova
...	...	...	...
72	184	67	Laughing Lumberjack Lager
73	138	48	Chocolade
74	125	37	Gravad lax
75	122	15	Genen Shouyu
76	95	9	Mishi Kobe Niku

77 rows × 3 columns

Can we filter the above query by year? Most sold products by year.

In [105]:

```
cur.execute('''
select sum(quantity) as total_sold, p.productid, productname,  extract(year from orderdate) as order_year
from n_products p
join n_order_details od on od.productid = p.productid
join n_orders o on o.orderid = od.orderid
group by p.productid, productname, order_year
order by order_year, total_sold desc
''')
out()
```

Out[105]:

	0	1	2	3
0	444	31	Gorgonzola Telino	1996.0
1	370	60	Camembert Pierrot	1996.0
2	274	35	Steeleye Stout	1996.0
3	266	39	Chartreuse verte	1996.0
4	261	71	Fløtemysost	1996.0
...	...	...	...	...
222	21	14	Tofu	1998.0
223	12	73	Röd Kaviar	1998.0
224	8	48	Chocolade	1998.0
225	3	9	Mishi Kobe Niku	1998.0
226	1	66	Louisiana Hot Spiced Okra	1998.0

227 rows × 4 columns

What is the business' Total Revenue by year. Show only a single row for each year.

In [106]:

```
cur.execute('''
select sum((quantity* unitprice) * (1-discount)) as total_revenue, extract(year from orderdate) as order_year
from n_orders o
join n_order_details od on od.orderid = o.orderid
group by order_year
order by order_year
''')
out()
```

Out[106]:

```
               0       1
0  208083.969755  1996.0
1  617085.202870  1997.0
2  440623.865663  1998.0
```

We would like to know the most popular destination (country) for our orders over the entire time the business has been operating. Show a list in descending order.

In [107]:

```
cur.execute('''
select count(orderid) as number_of_orders, country
from n_orders o
join n_customers c on o.customerid = c.customerid
group by country
order by number_of_orders desc
limit 5
''')
out()
```

Out[107]:

```
     0        1
0  122  Germany
1  122      USA
2   83   Brazil
3   77   France
4   56       UK
```

In [108]:

```
conn = psycopg2.connect(host="localhost", port = 5432, database="Mobile_price", user="postgres", password="calpoly")
cur = conn.cursor()
def out():
    query_results = pd.DataFrame(cur.fetchall())
    return query_results
```

Create a query that calculates the average subsidy as a percentage of the retail price for each manufacturer. Note: if a phone has a retail price of 400 and a subsidized price of 150, then the carrier and/or manufacturer have contributed a subsidy of 250 and this is 62.5% of the phones retail price

In [109]:

```
cur.execute('''
select avg(((retail_price - sub_price)/retail_price)*100) as subsidy, m.m_name
from d_device d
join d_device_price_warehouse dp on dp.device_id = d.device_id
join d_manufacturer m on m.manufacturer_id = d.manufacturer_id
where retail_price is not null and sub_price is not null
group by m.m_name
order by subsidy desc
limit 5
''')
out()
```

Out[109]:

```
            0           1
0  100.000000    Audiovox
1  100.000000       Sagem
2   93.335185     Firefly
3   89.967731  UT Starcom
4   82.100944     Pantech
```

Create a query that shows how many new devices each manufacturer has introduced in a two-year period from 2004 to 2005. For the query you will use the field "start_year" in table "d_device" to screen out only the devices that "started" in years 2004 or 2005

In [110]:

```
cur.execute('''
select count(m.manufacturer_id) as qty, m_name
from d_device d
join d_manufacturer m on m.manufacturer_id = d.manufacturer_id
where start_year between 2004 and 2005
group by m_name
order by qty desc
limit 5
''')
out()
```

Out[110]:

	0	1
0	101	Motorola
1	51	Samsung
2	47	LG
3	44	Nokia
4	19	Kyocera

. Produce a list of average device weights (dim_weight field) for each manufacturer. Include only years 2006 and 2007 (start_year field).

In [111]:

```
cur.execute('''
select avg(dim_weight) as AvgOfdim_weight, m_name as name, count(d.device_id) as No_Device
from d_device d
join d_manufacturer m on m.manufacturer_id = d.manufacturer_id
where start_year between 2006 and 2007
group by m_name
order by No_Device Desc, m_name
limit 5
''')
out()
```

Out[111]:

	0	1	2
0	86.122449	Samsung	101
1	99.588235	Motorola	51
2	99.878049	LG	41
3	106.911765	Nokia	35
4	87.941176	Sagem	17

In [112]:

```
conn = psycopg2.connect(host="localhost", port = 5432, database="Library", user="postgres", pas
sword="calpoly")
cur = conn.cursor()
def out():
    query_results = pd.DataFrame(cur.fetchall())
    return query_results
```

In [113]:

```
# Make a new table showing members (member_id, f_name, l_name) that have checked out books in
year 2015.
```

In [114]:

```
cur.execute('''
create table check_out_books_in_2016
as
select distinct m.member_id, f_name, l_name
from members m
join checkout_log cl on cl.member_id = m.member_id
where check_out_date  between '1/1/2016' and '12/31/2016';

select * from check_out_books_in_2016
limit 5
''')
out()
```

Out[114]:

	0	1	2
0	8	Bob	Doe
1	6	Gunther	Johnson
2	9	Jaqueline	Seiver
3	7	John	Mcquaid
4	14	Frank	Gorman

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Sub Quesry

Show titles of all movies generating more than average profit

(assume: p = gross_usa - budget) that also star actress "Julia Roberts"

In [5]:

```python
conn = psycopg2.connect(host="localhost", port = 5432, database="IMDB", user="postgres", password="calpoly")
cur = conn.cursor()
def out():
    query_results = pd.DataFrame(cur.fetchall())
    return query_results
```

In [116]:

```python
##get average by creating sub query
cur.execute(
'''select avg(gross_usa - budget) as profit
from imdb_movie m
join imdb_actor_movie_map am on am.movie_id = m.movie_id
join imdb_actor a on a.actor_id = am.actor_id
where actor_name = 'Julia Roberts'
and budget is not null and gross_usa is not null''')
out()
```

Out[116]:

```
            0
0  40294663.15
```

In [117]:

```python
#insurt average
cur.execute('''
select title, (gross_usa - budget) as profit
from imdb_movie m
join imdb_actor_movie_map am on am.movie_id = m.movie_id
join imdb_actor a on a.actor_id = am.actor_id
where actor_name = 'Julia Roberts'
and budget is not null and gross_usa is not null
and (gross_usa - budget) >
(
    -- compute avg for JR movie
    select avg(gross_usa - budget) as profit
    from imdb_movie m
    join imdb_actor_movie_map am on am.movie_id = m.movie_id
    join imdb_actor a on a.actor_id = am.actor_id
    where actor_name = 'Julia Roberts'
    and budget is not null and gross_usa is not null
)
''')
out()
```

Out[117]:

```
                         0            1
0                     Hook   49654823.0
1             Notting Hill   74006080.0
2          Valentine's Day   58476776.0
3           Ocean's Eleven   98405771.0
4           Erin Brockovich   73548685.0
5             Pretty Woman  164406268.0
6                  Stepmom   41137662.0
7  My Best Friend's Wedding   80805112.0
8   Sleeping with the Enemy   82599005.0
```

Different way

In [11]:

```python
cur.execute("""
    select avg(gross_usa - budget) as prof
    from imdb_movie
    where budget is not null and gross_usa is not null
""")
out()
```

Out[11]:

```
              0
0  1.987662e+07
```

In [119]:

```
cur.execute('''
select title, (gross_usa - budget) as profit
from imdb_movie m
join imdb_actor_movie_map am on  m.movie_id = am.movie_id
join imdb_actor a on a.actor_id = am.actor_id
where actor_name = 'Julia Roberts'
and (gross_usa - budget) >
(
    select avg(gross_usa - budget) as prof
    from imdb_movie
    where budget is not null and gross_usa is not null
)
''')
out()
```

Out[119]:

```
                           0            1
 0                      Hook   49654823.0
 1              Notting Hill   74006080.0
 2           Valentine's Day   58476776.0
 3            Ocean's Eleven   98405771.0
 4           Erin Brockovich   73548685.0
 5             Eat Pray Love   20574010.0
 6                Flatliners   35490000.0
 7             Pretty Woman   164406268.0
 8               The Mexican   32808615.0
 9                  Stepmom   41137662.0
10  My Best Friend's Wedding   80805112.0
11    Sleeping with the Enemy  82599005.0
```

Show titles of movies by directors having at least 10 or more movies.Also list the director's name and the movie's profit (assume: p = gross_usa - budget)

In [120]:

```
cur.execute('''
select d.director_id, director_name,count(movie_id) as no_movie
from imdb_movie m
join imdb_director d on m.director_id = d.director_id
group by director_name, d.director_id
order by no_movie desc
''')
out()
```

Out[120]:

```
         0                 1   2
   0   207  Steven Spielberg  27
   1    10    Clint Eastwood  21
   2   212      Ridley Scott  19
   3    32       Ron Howard   18
   4   461        Tim Burton  18
 ...   ...               ...  ...
1901  1343    Justin Benson    1
1902  1252     Orson Welles    1
1903  1785   Ali Abbas Zafar   1
1904  1323       Nick Cannon   1
1905  1236    Sharon Maguire   1

1906 rows × 3 columns
```

In [121]:

```
cur.execute('''
select title,director_name, (gross_usa - budget) as profit
from imdb_director d
join imdb_movie m  on m.director_id = d.director_id
where budget is not null and gross_usa is not null
and d.director_id in
(
    select director_id from
    (
        select d.director_id, director_name,count(movie_id) as no_movie
        from imdb_movie m
        join imdb_director d on m.director_id = d.director_id
        group by director_name, d.director_id
        order by no_movie desc
    ) as ctx
    where no_movie >= 10
)
''')
out()
```

Out[121]:

	0	1	2
0	Unforgiven	Clint Eastwood	60281912.0
1	Back to the Future Part III	Robert Zemeckis	47727583.0
2	The Quick and the Dead	Sam Raimi	-13363463.0
3	The Missing	Ron Howard	-33099664.0
4	The Outlaw Josey Wales	Clint Eastwood	28100000.0
...	...	...	...
360	The Ghost Writer	Roman Polanski	-29458451.0
361	Phone Booth	Joel Schumacher	33566212.0
362	The Ninth Gate	Roman Polanski	-19346254.0
363	Space Cowboys	Clint Eastwood	25454043.0
364	Village of the Damned	John Carpenter	-12582433.0

365 rows × 3 columns

Epics Those exceptionally long movies... Write a query that identifies movies that have "run_times" longer than 99 % of all movies and having budgets over

In [122]:

```
cur.execute('''
select percentile_cont(0.99) within group (order by run_time)
from imdb_movie
''')
out()
```

Out[122]:

	0
0	180.0

In [10]:

```
cur.execute('''
select title, budget
from imdb_movie
where budget > 10000000
and run_time > (
    select percentile_cont(0.99) within group (order by run_time)
    from imdb_movie
)
limit 5
''')
out()
```

Out[10]:

	0	1
0	Dances with Wolves	19000000.0
1	Wyatt Earp	63000000.0
2	Heaven's Gate	44000000.0
3	Pearl Harbor	140000000.0
4	The Deer Hunter	15000000.0

In [7]:

```
cur.execute('''
select title, user_rating,budget, gross_usa
from imdb_movie
where budget > 20000000
and user_rating < (
      select percentile_cont(0.75) within group (order by user_rating)
      from imdb_movie
)
and gross_usa < (budget*0.25)
order by user_rating
limit 5
''')
out()
```

Out[7]:

	0	1	2	3
0	Son of the Mask	2.1	84000000.0	17010646.0
1	Gigli	2.3	54000000.0	6068735.0
2	Dragonball: Evolution	2.8	45000000.0	9362785.0
3	The Adventures of Pluto Nash	3.7	100000000.0	4420080.0
4	In the Name of the King: A Dungeon Siege Tale	3.8	60000000.0	4535117.0

Week8 State Finder

In [19]:

```
conn = psycopg2.connect(host="localhost", port = 5432, database="StateFinder", user="postgres"
, password="calpoly")
cur = conn.cursor()
def out():
   query_results = pd.DataFrame(cur.fetchall())
   return query_results
```

In [21]:

```
cur.execute('''
select s.state_ID, state_name, (house_value_2017/house_value_2010) as change_in_hv
from states s
join states_avg_house_value_3bed tb on s.state_ID = tb.state_ID
order by change_in_hv desc
limit 5
''')
out()
```

Out[21]:

	0	1	2
0	NV	Nevada	1.532632
1	CO	Colorado	1.523765
2	ND	North Dakota	1.484626
3	CA	California	1.477955
4	MI	Michigan	1.422460

Find states in the bottom quartile for average house value increase (from 2010 to 2017) and in the top quartile for increase in median household income (2013 to 2017).

In [22]:

```
cur.execute('''
select s.state_id, state_name
from states s
join states_avg_house_value_3bed tb on s.state_ID = tb.state_ID
join states_median_household_income hi on s.state_ID = hi.state_ID
where (house_value_2017/house_value_2010) <=
(-- cut off value for bottom quantile delta hv
        select percentile_cont(0.25) within group(order by change_in_hv)
        from(-- list of all state and the their change in house value
                select s.state_ID, state_name, (house_value_2017/house_value_2010) as change_in_hv
                from states s
                join states_avg_house_value_3bed tb on s.state_ID = tb.state_ID
                order by change_in_hv desc
        ) as delta_hv
)
-- list of state in top quatile delta hi
and (median_2017/median_2013) >=
(-- cut off value for top quantile delta hv
        select percentile_cont(0.75) within group(order by change_in_hi)
        from(-- list of all state and the their change in household income
                select s.state_ID, state_name, (median_2017/median_2013) as change_in_hi
                from states s
                join states_median_household_income hi on s.state_ID = hi.state_ID
                order by change_in_hi desc
        ) as delta_hi
)
''')
out()
```

Out[22]:

	0	1
0	AZ	Arizona
1	WI	Wisconsin

Find states in the top quartile for per-capita GDP (gdp_2018 / pop_2018)* note GDP in this data set are in millions-of-dollars, so multiply this number by 1,000,000 to get the GDP number into dollars)

In [23]:

```
cur.execute('''

select state_name, ((gdp_2018*1000000)/pop_2018) as per_chapita_GDP
from states s
join states_gdp_current_dollar gdp on s.state_id = gdp.state_id
join states_population p on p.state_id = s.state_id
where ((gdp_2018*1000000)/pop_2018) >=
--cutoff value
(
        select percentile_cont(0.75) within group(order by per_chapita_GDP)
        from
        (
                select ((gdp_2018*1000000)/pop_2018) as per_chapita_GDP
                from states s
                join states_gdp_current_dollar gdp on s.state_id = gdp.state_id
                join states_population p on p.state_id = s.state_id
                order by per_chapita_GDP desc
        ) as gdp
) order by per_chapita_GDP desc
limit 5
''')
out()
```

Out[23]:

	0	1
0	New York	85398.022301
1	Massachusetts	82508.795449
2	Connecticut	77176.841378
3	Delaware	75975.189496
4	California	75782.531279

Find states with below-average change in house_value, below-average change in population, and above-average yearly_clear_days, and above-average change in median_household_income.

In [25]:

```
cur.execute('''

--q6
select state_name, state_id
from states
where state_id in
(
    -- list of below average delta hv
    select s.state_id
    from states s
    join states_avg_house_value_3bed tb on s.state_ID = tb.state_ID
    where (house_value_2017/house_value_2010) <=
    (-- cut off value for bottom quantile delta hv
            select avg(house_value_2017/house_value_2010) as avg_change_in_hv
            from states s
            join states_avg_house_value_3bed tb on s.state_ID = tb.state_ID
    )
)
and state_id in
(
    -- list of below average delta population
    select s.state_id
    from states s
    join states_population p on s.state_ID = p.state_ID
    where (pop_2018/pop_2010) <=
    (-- cut off value for bottom quantile delta hv
            select avg(pop_2018/pop_2010) as avg_change_in_pop
            from states s
            join states_population p on s.state_ID = p.state_ID
    )
)
and state_id in
(
    -- list of above average clear days
    select s.state_id
    from states s
    join states_sun ss  on s.state_id = ss.state_id
    where yearly_clear_days >
    (
        select avg(yearly_clear_days) as avg_sun
        from states s
        join states_sun ss  on s.state_id = ss.state_id
    )
)
and state_id in
(
    --list of delta household income
    select s.state_id
    from states s
    join states_median_household_income hi on s.state_ID = hi.state_ID
    where (median_2017/median_2013) >
    (
        select avg(median_2017/median_2013) as avg_change_hi
        from states s
        join states_median_household_income hi on s.state_ID = hi.state_ID
        order by avg_change_hi desc
    )
)
''')
out()
```

Out[25]:

```
            0    1
0    Arkansas   AR
1 Mississippi   MS
2    Missouri   MO
```

In [5]:

```
conn = psycopg2.connect(host="localhost", port = 5432, database="MLB", user="postgres", passw
ord="calpoly")
cur = conn.cursor()
def out():
   query_results = pd.DataFrame(cur.fetchall())
   return query_results
```

Order this query’s output descending by the number of teams played for. Four pitchers are tied sharing the title as playing for the most different teams. List all four pitchers’ names. Also write the producing SQL query.

In [6]:

```
cur.execute('''
select count(distinct team_id) as num_team, ps.player_id, player_name
from mlb_players_seasons ps
join mlb_players p on p.player_id = ps.player_id
group by player_name, ps.player_id
order by num_team desc
limit 4
''')
out()
```

Out[6]:

```
    0       1            2
0  11  112526     Colon, B
1  11  119374    Morgan, M
2  11  110683   Batista, M
3  11  123725   Villone, R
```

Write a query that locates pitchers that have played for “Seattle” AND “Arizona” at one point in their careers. Create a column showing the total “innings pitched” for each player and sort this in descending order. Who are the top three players?

In [9]:

```
cur.execute('''
select player_name, ps.player_id, sum(innings_pitched) as total_innings
from mlb_players_seasons ps
join mlb_players p on p.player_id = ps.player_id
where ps.player_id in
(
    select player_id from mlb_players_seasons ps
    join mlb_teams t on t.team_id = ps.team_id
    where team_name like '%Seattle%'
)
and ps.player_id in
(
    select player_id from mlb_players_seasons ps
    join mlb_teams t on t.team_id = ps.team_id
    where team_name like '%Arizona%'
)
group by player_name, ps.player_id
order by total_innings desc
limit 5
''')
out()
```

Out[9]:

	0	1	2
0	Johnson, R	116615	4130.2
1	Morgan, M	119374	2767.9
2	Mulholland, T	119488	2571.7
3	Benes, A	110854	2502.3
4	Hampton, M	115399	2265.3

For this query we’re going to locate pitchers that have experienced both outstanding “winning” seasons as well as terrifying “losing” seasons. Locate pitchers that have had seasons where their wins are greater-than-or-equal-to the 99.9th percentile (0.999) of all wins in the dataset. Further constrain this set to pitchers that have also experienced seasons where their losses are greater-than-or-equal-to the 99.9th percentile (0.999) of all losses in the dataset. Order the query by total wins descending

In [10]:

```
cur.execute('''
/*good season, bad season*/
select player_name, ps.player_id, sum(wins) total_wins, sum(losses)
from mlb_players_seasons ps
join mlb_players p on p.player_id = ps.player_id
where ps.player_id in
(   /* great season */
    select player_id from mlb_players_seasons ps
    where wins >=
    (
        select percentile_cont(0.999) within group (order by wins)
        from mlb_players_seasons
    )
)
and ps.player_id in
(   /* horrible season */
    select player_id from mlb_players_seasons ps
    where losses >=
    (
        select percentile_cont(0.999) within group (order by losses)
        from mlb_players_seasons
    )
)
group by player_name, ps.player_id
order by total_wins desc
''')
out()
```

Out[10]:

	0	1	2	3
0	Carlton, S	112008	329	244
1	Lolich, M	117875	217	191
2	Jackson, L	116434	194	183
3	Wood, W	124543	164	156
4	McLain, D	118787	131	91

Among this 300 wins club, which pitcher has lowest average “walks_allowed” / “game” ratio?

In [11]:

```
cur.execute('''
select avg(walks_allowed/games) as average_walk_per_game, p.player_id, player_name
from mlb_players_seasons ps
join mlb_players p on p.player_id = ps.player_id
where p.player_id in
(
    select player_id
    from
    (
        select player_id, sum(wins) as sum_wins
        from mlb_players_seasons
        group by player_id
    ) as table_sum_wins
    where sum_wins >= 300
)
group by p.player_id, player_name
order by average_walk_per_game
limit 5
''')
out()
```

Out[11]:

```
                         0        1          2
0   0.91304347826086956522   118120  Maddux, G
1       1.2173913043478261   123006   Sutton, D
2       1.2272727272727273   120438    Perry, G
3       1.6250000000000000   119786   Niekro, P
4       1.7000000000000000   121961   Seaver, T
```

Make a query that lists the teams having the most seasons with pitchers that have won 300 games or more.

In [12]:

```
cur.execute('''
select distinct(t.team_id), count(t.team_id) as total_num_games, team_name
from mlb_players_seasons ps
join mlb_players p on p.player_id = ps.player_id
join mlb_teams t on t.team_id = ps.team_id
where p.player_id in
(
    select player_id
    from
    (
        select player_id, sum(wins) as sum_wins
        from mlb_players_seasons
        group by player_id
    ) as table_sum_wins
    where sum_wins >= 300
)
group by t.team_id, team_name
order by total_num_games desc
limit 5
''')
out()
```

Out[12]:

```
    0   1                    2
0   3  48       Atlanta Braves
1  28  21        New York Mets
2  21  18  Los Angeles Dodgers
3   5  14      Boston Red Sox
4  16  14       Houston Astros
```

Identify the most losing team. Use this as a subquery to feed back into a higher-level query that identifies the most winning (sum(wins)) pitcher on the most losing team.

In [13]:

```
cur.execute('''
select sum(wins) as wins, p.player_id, player_name, team_name
from mlb_players_seasons ps
join mlb_players p on p.player_id = ps.player_id
join mlb_teams t on t.team_id = ps.team_id
where t.team_id in
(
	select team_id
	from
	(
		select t.team_id, team_name, avg(wins/losses) as wl_ratio
		from mlb_teams_seasons ts
		join mlb_teams t on ts.team_id = t.team_id
		group by t.team_id,team_name
		order by wl_ratio
		limit 1
	) as table_most_lost_team
)
group by p.player_id, player_name, team_name
order by wins desc
limit 5''')
out()
```

Out[13]:

```
     0       1            2              3
0   38  605228  Fernandez, J  Miami Marlins
1   35  543408    Koehler, T  Miami Marlins
2   28  570632      Urena, J  Miami Marlins
3   23  543045     Conley, A  Miami Marlins
4   21  445197       Dunn, M  Miami Marlins
```

In [ ]: