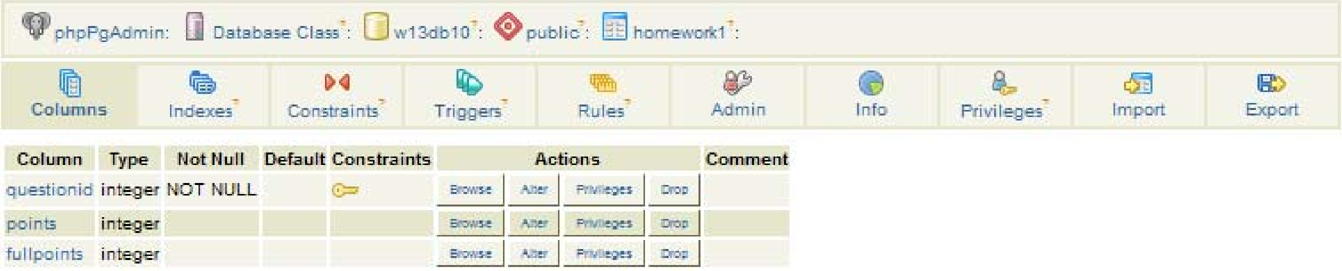
Homework 1

Name: Dejun Qian
PSUID: 917233450

Question 1:
Demonstrate the difference between a primary key and a “unique” declaration for an attribute in a table in Postgresql with regard to how they handle null values.
Answer:
The following SQL query creates a table with a key attribute.

```
CREATE TABLE homework1 (questionid INT,
                        points INT,
                        fullpoints INT,
                        PRIMARY KEY(questionid))
```

The screenshot of the result is given bellow. We can see “questionid” is set as the primary key.

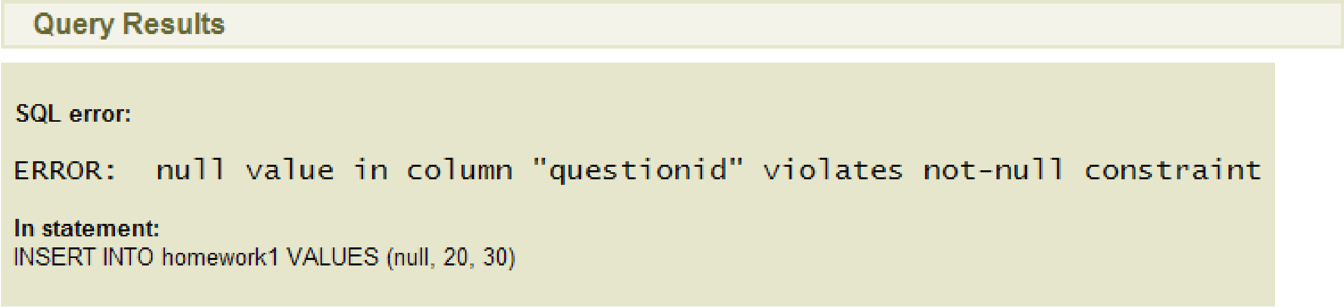


Column	Type	Not Null	Default	Constraints	Actions				Comment
questionid	integer	NOT NULL		🔑	Browse	Alter	Privileges	Drop	
points	integer				Browse	Alter	Privileges	Drop	
fullpoints	integer				Browse	Alter	Privileges	Drop	

The following SQL query is used to add a null value.

```
INSERT INTO homework1
VALUES (null, 20, 30)
```

The result for this query is given bellow, which shows we are not able to have null value for key attribute.


Query Results

SQL error:

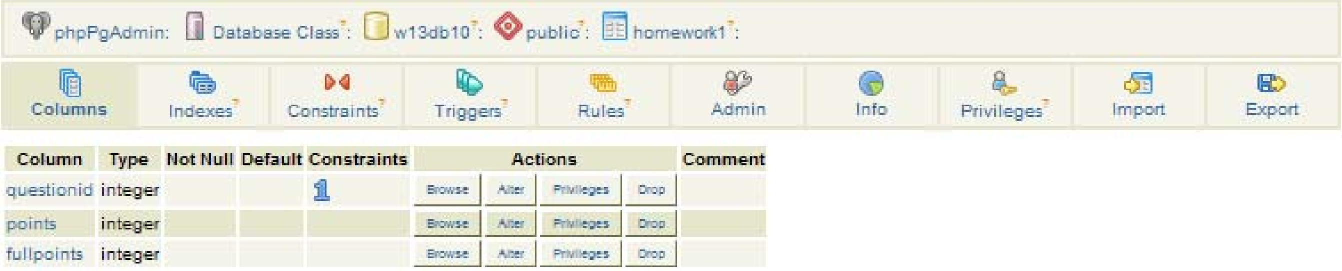
```
ERROR:  null value in column "questionid" violates not-null constraint
```

In statement:
INSERT INTO homework1 VALUES (null, 20, 30)

In contrast, we issue the following query to create a table with a unique attribute.

```
CREATE TABLE homework1 (questionid INT,
                        points INT,
                        fullpoints INT,
                        UNIQUE(questionid))
```

The screenshot of the result is shown bellow.

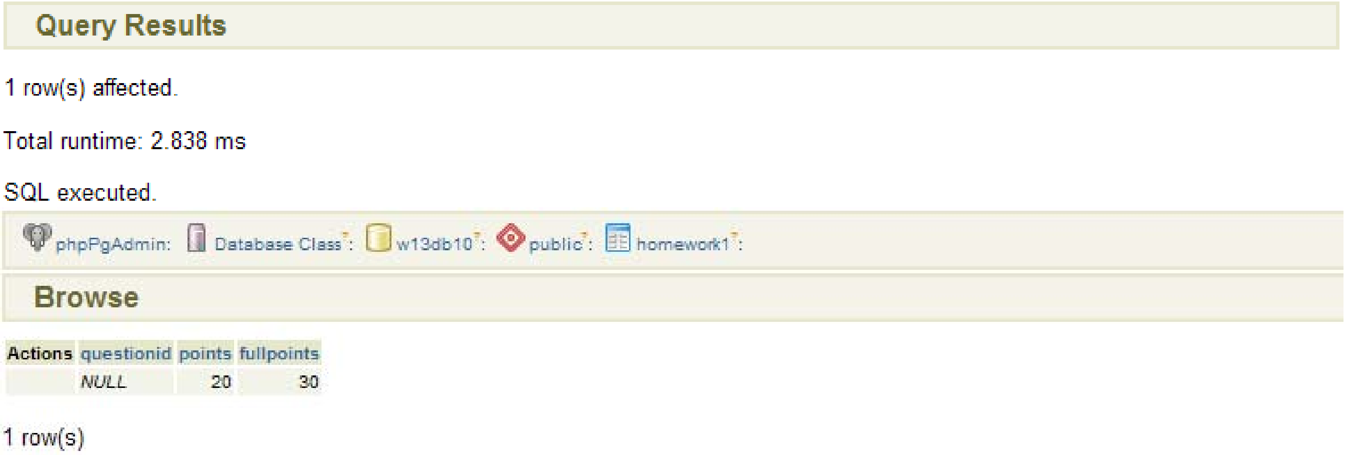


Column	Type	Not Null	Default	Constraints	Actions				Comment
questionid	integer			1	Browse	Alter	Privileges	Drop	
points	integer				Browse	Alter	Privileges	Drop	
fullpoints	integer				Browse	Alter	Privileges	Drop	

The following query which is used to add a null value is successfully executed.

```
INSERT INTO homework1
VALUES (null, 20, 30)
```

The result is shown bellow.



Query Results

1 row(s) affected.

Total runtime: 2.838 ms

SQL executed.

Browse

Actions	questionid	points	fullpoints
	NULL	20	30

1 row(s)

Question 2:

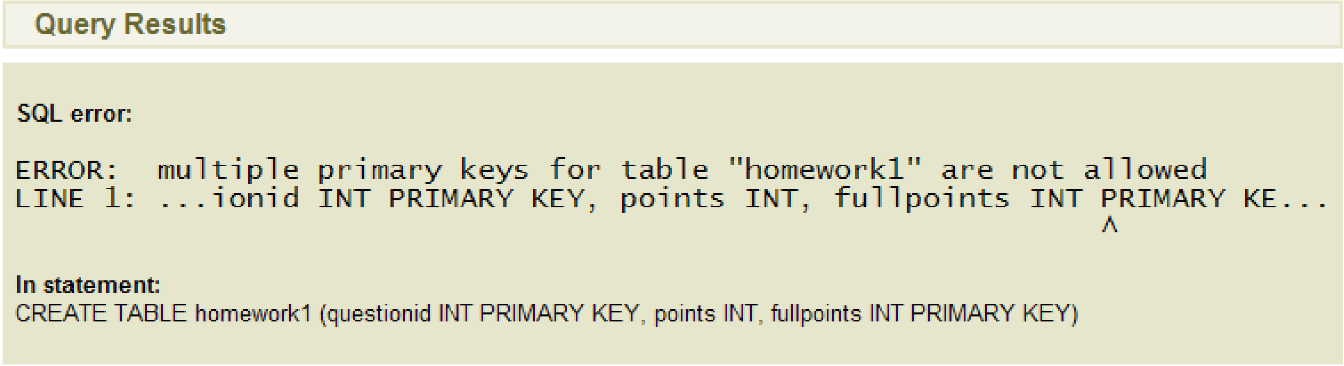
Demonstrate that you can (or cannot) have two primary keys for one table. Includes screenshots as appropriate.

Answer:

We can’t have a table with two primary keys. The following query is used to try this job.

```
CREATE TABLE homework1 (questionid INT PRIMARY KEY,
                        points INT,
                        fullpoints INT PRIMARY KEY)
```

The result is shown bellow which gives the error message.



Query Results

SQL error:

```
ERROR:  multiple primary keys for table "homework1" are not allowed
LINE 1: ...ionid INT PRIMARY KEY, points INT, fullpoints INT PRIMARY KE...
                                                             ^
```

In statement:
CREATE TABLE homework1 (questionid INT PRIMARY KEY, points INT, fullpoints INT PRIMARY KEY)

Question 3:

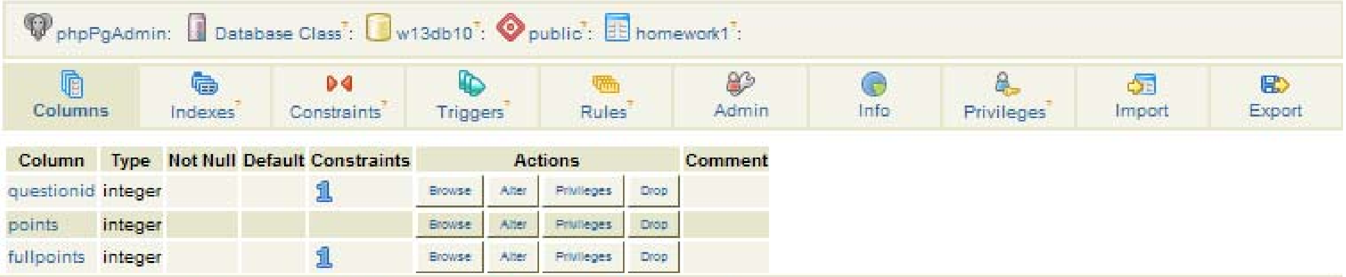
Demonstrate that you can (or cannot) have two different attributes declared as unique for one table. Includes screenshots as appropriate.

Answer:

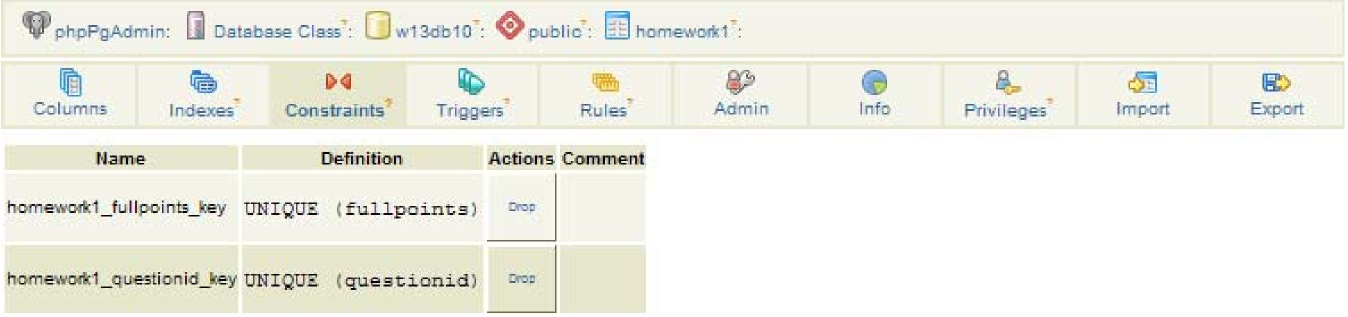
We can have a table with two different unique attributes. The following query creates a table of this kind.

```
CREATE TABLE homework1 (questionid INT UNIQUE,
                        points INT,
                        fullpoints INT UNIQUE)
```

The result of the above query is given bellow.



Column	Type	Not Null	Default	Constraints	Actions				Comment
questionid	integer			1	Browse	Alter	Privileges	Drop	
points	integer				Browse	Alter	Privileges	Drop	
fullpoints	integer			1	Browse	Alter	Privileges	Drop	



Name	Definition	Actions	Comment
homework1_fullpoints_key	UNIQUE (fullpoints)	Drop	
homework1_questionid_key	UNIQUE (questionid)	Drop	

Question 4:

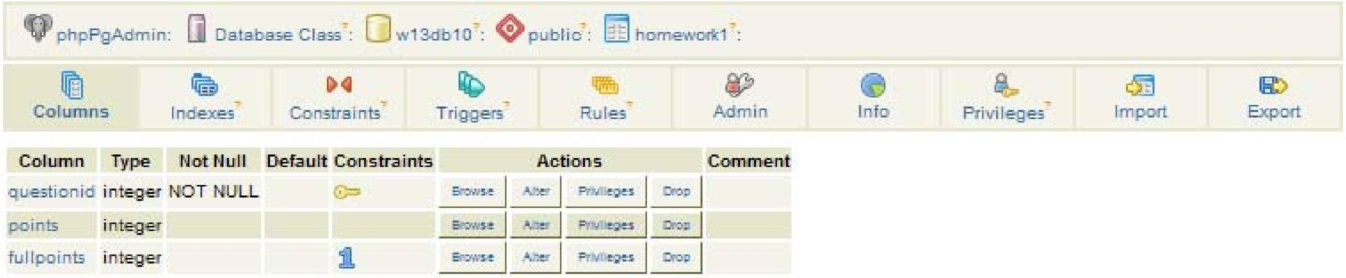
Demonstrate that you can (or cannot) have one attribute as a primary key and another attribute as unique for one table. Includes screenshots as appropriate.

Answer:

We can have one attribute as a primary key and another as unique for one table. The following query does this job.

```
CREATE TABLE homework1 (questionid INT PRIMARY KEY,
                        points INT,
                        fullpoints INT UNIQUE)
```

The result after execution of this query is shown bellow.



Column	Type	Not Null	Default	Constraints	Actions				Comment
questionid	integer	NOT NULL		🔑	Browse	Alter	Privileges	Drop	
points	integer				Browse	Alter	Privileges	Drop	
fullpoints	integer			1	Browse	Alter	Privileges	Drop	

Question 5:

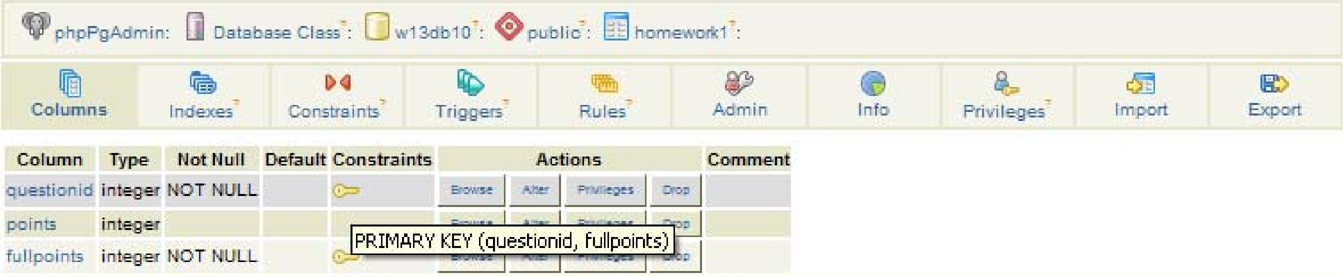
Write and execute an SQL statement that creates a table with a key that consists of two attributes. Show the SQL statement that you used. Show a screenshot that shows the result after you execute this SQL statement.

Answer:

The following SQL statement is used to create a table with a key that consists of two attributes.

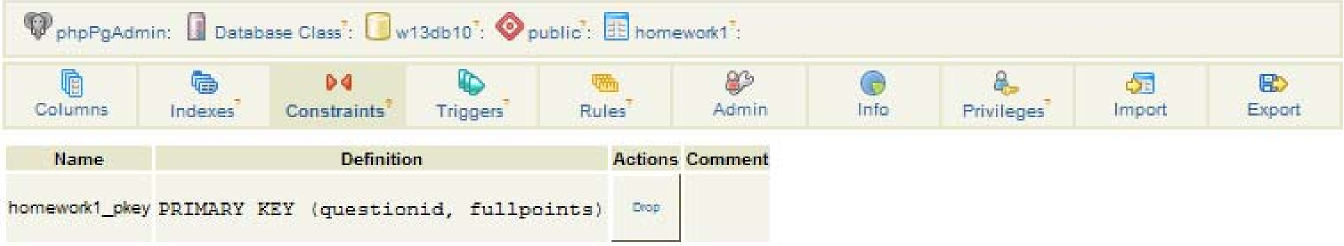
```
CREATE TABLE homework1 (questionid INT,
                        points INT,
                        fullpoints INT,
                        PRIMARY KEY(questionid, fullpoints))
```

The screenshot of the result is shown bellow.



Column	Type	Not Null	Default	Constraints	Actions				Comment
questionid	integer	NOT NULL		🔑	Browse	Alter	Privileges	Drop	
points	integer				Browse	Alter	Privileges	Drop	
fullpoints	integer	NOT NULL		🔑	Browse	Alter	Privileges	Drop	

PRIMARY KEY (questionid, fullpoints)

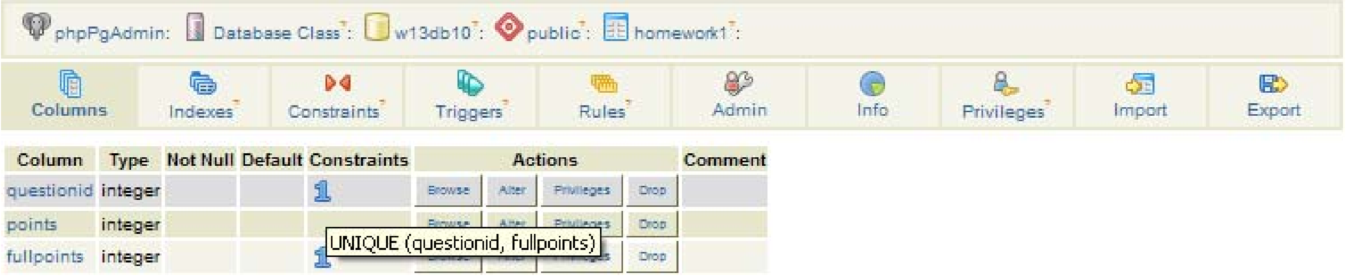


Name	Definition	Actions	Comment
homework1_pkey	PRIMARY KEY (questionid, fullpoints)	Drop	

I also tested this for unique. The statement is shown bellow.

```
CREATE TABLE homework1 (questionid INT,
                        points INT,
                        fullpoints INT,
                        UNIQUE(questionid, fullpoints))
```

The following is the result.

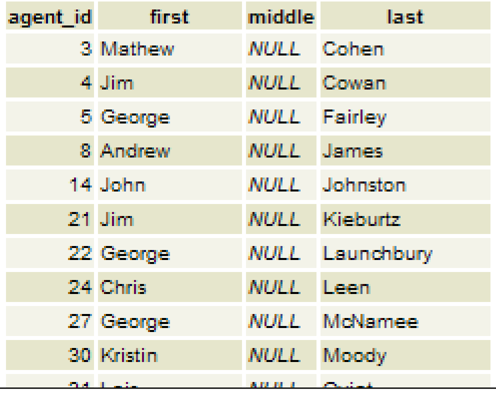


Column	Type	Not Null	Default	Constraints	Actions				Comment
questionid	integer			1	Browse	Alter	Privileges	Drop	
points	integer				Browse	Alter	Privileges	Drop	
fullpoints	integer			1	Browse	Alter	Privileges	Drop	

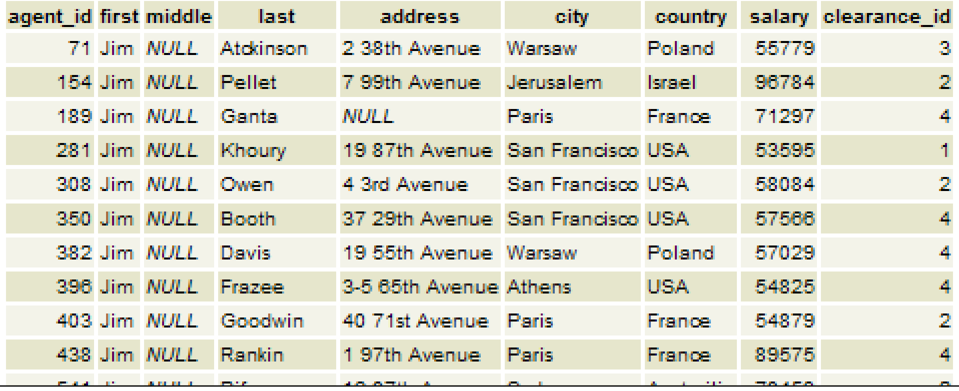
UNIQUE (questionid, fullpoints)

The result here looks like the result from question 3. However, they are actually different. In question 3, you can't have duplicated value for each of the attribute, which you can have some duplicated values as long as not all of the attributes have the same value.

Question 6:
List the agent_id, agent first, middle, and last for agents with a salary greater than 52000.
Answer:

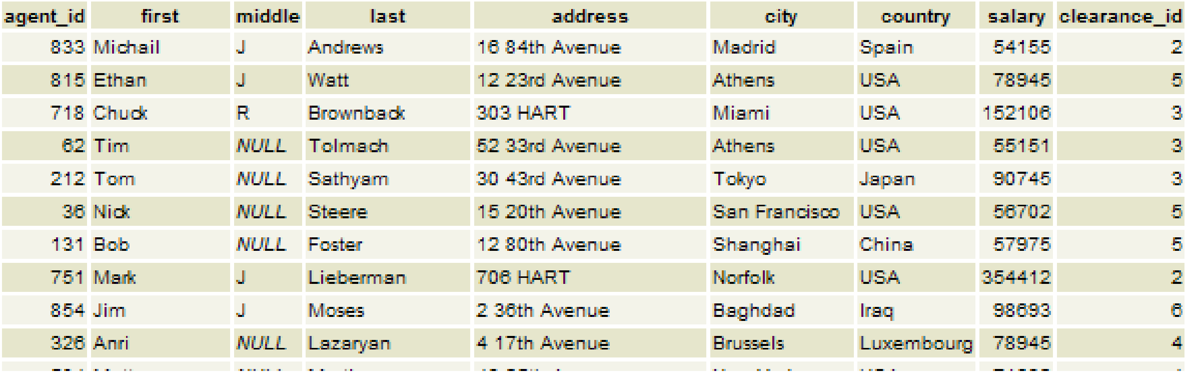
<table>
<tr><td>SQL query:</td><td>SELECT agent_id, first, middle, last
FROM agent
WHERE salary &gt; 52000</td></tr>
<tr><td>Number of rows:</td><td>584 row(s)</td></tr>
<tr><td>The first 10 rows:</td><td>
<table>
<tr><th>agent_id</th><th>first</th><th>middle</th><th>last</th></tr>
<tr><td>3</td><td>Mathew</td><td><i>NULL</i></td><td>Cohen</td></tr>
<tr><td>4</td><td>Jim</td><td><i>NULL</i></td><td>Cowan</td></tr>
<tr><td>5</td><td>George</td><td><i>NULL</i></td><td>Fairley</td></tr>
<tr><td>8</td><td>Andrew</td><td><i>NULL</i></td><td>James</td></tr>
<tr><td>14</td><td>John</td><td><i>NULL</i></td><td>Johnston</td></tr>
<tr><td>21</td><td>Jim</td><td><i>NULL</i></td><td>Kieburtz</td></tr>
<tr><td>22</td><td>George</td><td><i>NULL</i></td><td>Launchbury</td></tr>
<tr><td>24</td><td>Chris</td><td><i>NULL</i></td><td>Leen</td></tr>
<tr><td>27</td><td>George</td><td><i>NULL</i></td><td>McNamee</td></tr>
<tr><td>30</td><td>Kristin</td><td><i>NULL</i></td><td>Moody</td></tr>
</table></td></tr>
<tr><td>Relational algebra:</td><td>$\pi_{agent\_id,first,middle,last}(\sigma_{salary>52000}(agent))$</td></tr>
</table>

Question 7:
List all attributes for agents with a first name of Jim who have a security clearance less than 5.
Answer:

<table>
<tr><td>SQL query:</td><td>SELECT *
FROM agent
WHERE first = 'Jim' AND clearance_id &lt; 5</td></tr>
<tr><td>Number of rows:</td><td>15 row(s)</td></tr>
<tr><td>The first 10 rows:</td><td>
<table>
<tr><th>agent_id</th><th>first</th><th>middle</th><th>last</th><th>address</th><th>city</th><th>country</th><th>salary</th><th>clearance_id</th></tr>
<tr><td>71</td><td>Jim</td><td><i>NULL</i></td><td>Atckinson</td><td>2 38th Avenue</td><td>Warsaw</td><td>Poland</td><td>55779</td><td>3</td></tr>
<tr><td>154</td><td>Jim</td><td><i>NULL</i></td><td>Pellet</td><td>7 99th Avenue</td><td>Jerusalem</td><td>Israel</td><td>96784</td><td>2</td></tr>
<tr><td>189</td><td>Jim</td><td><i>NULL</i></td><td>Ganta</td><td><i>NULL</i></td><td>Paris</td><td>France</td><td>71297</td><td>4</td></tr>
<tr><td>281</td><td>Jim</td><td><i>NULL</i></td><td>Khoury</td><td>19 87th Avenue</td><td>San Francisco</td><td>USA</td><td>53595</td><td>1</td></tr>
<tr><td>308</td><td>Jim</td><td><i>NULL</i></td><td>Owen</td><td>4 3rd Avenue</td><td>San Francisco</td><td>USA</td><td>58084</td><td>2</td></tr>
<tr><td>350</td><td>Jim</td><td><i>NULL</i></td><td>Booth</td><td>37 29th Avenue</td><td>San Francisco</td><td>USA</td><td>57566</td><td>4</td></tr>
<tr><td>382</td><td>Jim</td><td><i>NULL</i></td><td>Davis</td><td>19 55th Avenue</td><td>Warsaw</td><td>Poland</td><td>57029</td><td>4</td></tr>
<tr><td>396</td><td>Jim</td><td><i>NULL</i></td><td>Frazee</td><td>3-5 65th Avenue</td><td>Athens</td><td>USA</td><td>54825</td><td>4</td></tr>
<tr><td>403</td><td>Jim</td><td><i>NULL</i></td><td>Goodwin</td><td>40 71st Avenue</td><td>Paris</td><td>France</td><td>54879</td><td>2</td></tr>
<tr><td>438</td><td>Jim</td><td><i>NULL</i></td><td>Rankin</td><td>1 97th Avenue</td><td>Paris</td><td>France</td><td>89575</td><td>4</td></tr>
</table></td></tr>
<tr><td>Relational algebra:</td><td>$\sigma_{first='Jim' \text{ AND } clearance\_id<5}(agent)$</td></tr>
</table>

Question 8:
List all attributes for agents that do NOT appear in the answer to query 7 just above. (Hint: use the EXCEPT clause in SQL.)
Answer:

SQL query:	(SELECT *
	FROM agent)
	EXCEPT
	(SELECT *
	FROM agent
	WHERE first = 'Jim' AND clearance_id < 5)

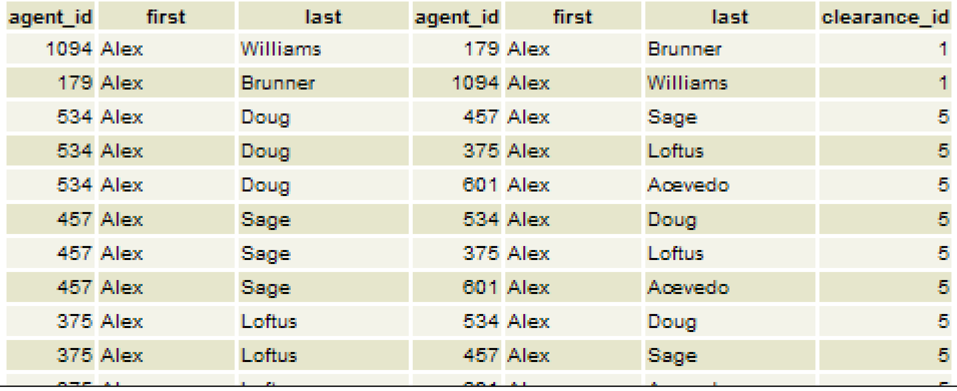
Number of rows:	647 row(s)
The first 10 rows:	
Relational algebra:	agent - $\sigma_{\text{first='Jim' AND clearance\_id<5}}$(agent)

agent_id	first	middle	last	address	city	country	salary	clearance_id
833	Michail	J	Andrews	16 84th Avenue	Madrid	Spain	54155	2
815	Ethan	J	Watt	12 23rd Avenue	Athens	USA	78945	5
718	Chuck	R	Brownback	303 HART	Miami	USA	152106	3
62	Tim	*NULL*	Tolmach	52 33rd Avenue	Athens	USA	55151	3
212	Tom	*NULL*	Sathyam	30 43rd Avenue	Tokyo	Japan	90745	3
36	Nick	*NULL*	Steere	15 20th Avenue	San Francisco	USA	56702	5
131	Bob	*NULL*	Foster	12 80th Avenue	Shanghai	China	57975	5
751	Mark	J	Lieberman	706 HART	Norfolk	USA	354412	2
854	Jim	J	Moses	2 36th Avenue	Baghdad	Iraq	98693	6
326	Anri	*NULL*	Lazaryan	4 17th Avenue	Brussels	Luxembourg	78945	4

Question 9:

List the two agent_ids, the two first and last names, and the security clearance for all pairs of agents where the two agents have the same first name, different last names, and the same security clearance.
How can you check to make sure that the rows in the query answer meet the above criteria?
How would you check (by issuing additional queries and examining the results) to see if there are any other agent pairs that meet the above criteria but did NOT appear in your query result?

Answer:

<table>
<tr><td>SQL query:</td><td>SELECT a1.agent_id,a1.first,a1.last,a2.agent_id,a2.first,a2.last,a1.clearance_id
FROM agent a1, agent a2
WHERE a1.first=a2.first AND a1.last!=a2.last AND a1.clearance_id=a2.clearance_id</td></tr>
<tr><td>Number of rows:</td><td>2172 row(s)</td></tr>
<tr><td>The first 10 rows:</td><td>
<table>
<tr><th>agent_id</th><th>first</th><th>last</th><th>agent_id</th><th>first</th><th>last</th><th>clearance_id</th></tr>
<tr><td>1094</td><td>Alex</td><td>Williams</td><td>179</td><td>Alex</td><td>Brunner</td><td>1</td></tr>
<tr><td>179</td><td>Alex</td><td>Brunner</td><td>1094</td><td>Alex</td><td>Williams</td><td>1</td></tr>
<tr><td>534</td><td>Alex</td><td>Doug</td><td>457</td><td>Alex</td><td>Sage</td><td>5</td></tr>
<tr><td>534</td><td>Alex</td><td>Doug</td><td>375</td><td>Alex</td><td>Loftus</td><td>5</td></tr>
<tr><td>534</td><td>Alex</td><td>Doug</td><td>601</td><td>Alex</td><td>Acevedo</td><td>5</td></tr>
<tr><td>457</td><td>Alex</td><td>Sage</td><td>534</td><td>Alex</td><td>Doug</td><td>5</td></tr>
<tr><td>457</td><td>Alex</td><td>Sage</td><td>375</td><td>Alex</td><td>Loftus</td><td>5</td></tr>
<tr><td>457</td><td>Alex</td><td>Sage</td><td>601</td><td>Alex</td><td>Acevedo</td><td>5</td></tr>
<tr><td>375</td><td>Alex</td><td>Loftus</td><td>534</td><td>Alex</td><td>Doug</td><td>5</td></tr>
<tr><td>375</td><td>Alex</td><td>Loftus</td><td>457</td><td>Alex</td><td>Sage</td><td>5</td></tr>
</table></td></tr>
<tr><td>Relational algebra:</td><td>$\pi_{a1.agent\_id,a1.first,a1.last,a2.agent\_id,a2.first,a2.last,a1.clearance\_id}($
$\sigma_{a1.first=a2.first\ AND\ a1.last!=a2.last\ AND\ a1.clearance\_id=a2.clearance\_id}(\rho_{a1}(agent)\times\rho_{a2}(agent))$
$)$</td></tr>
</table>

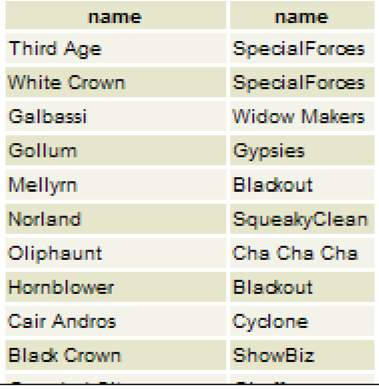
Let R denote the query answer, to check the rows in the query answer meet the criteria, we can check if $\sigma_{a1.first!=a2.first}(R)$ and $\sigma_{a1.last=a2.last}(R)$ are both null. If they are both null, then the result is OK.

To check if there are any other agent pairs that meet that above criteria but did NOT appear in the query result, we can first issue another query to get the complement of R. We denote the result as R1, and then we make sure there are no rows in R1 which meet the above criteria.

Question 10:

List the mission name and the team name where the team is assigned to the mission. Hint: use a cross product and a select and project operator in relational algebra. (Do something similar in SQL.)

Answer:

<table>
<tr><td>SQL query:</td><td>SELECT mission.name,team.name
FROM mission,team
WHERE mission.team_id=team.team_id</td></tr>
<tr><td>Number of rows:</td><td>404 row(s)</td></tr>
<tr><td>The first 10 rows:</td><td>
<table>
<tr><th>name</th><th>name</th></tr>
<tr><td>Third Age</td><td>SpecialForces</td></tr>
<tr><td>White Crown</td><td>SpecialForces</td></tr>
<tr><td>Galbassi</td><td>Widow Makers</td></tr>
<tr><td>Gollum</td><td>Gypsies</td></tr>
<tr><td>Mellyrn</td><td>Blackout</td></tr>
<tr><td>Norland</td><td>SqueakyClean</td></tr>
<tr><td>Oliphaunt</td><td>Cha Cha Cha</td></tr>
<tr><td>Hornblower</td><td>Blackout</td></tr>
<tr><td>Cair Andros</td><td>Cyclone</td></tr>
<tr><td>Black Crown</td><td>ShowBiz</td></tr>
</table></td></tr>
<tr><td>Relational algebra:</td><td>$\pi_{mission.name,team.name}(\sigma_{mission.team\_id=team.team\_id} (mission\times team))$</td></tr>
</table>

Question 11:

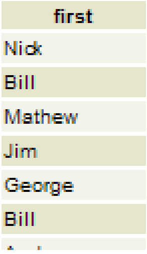
Write a query against the Spy database that demonstrates that SQL does NOT eliminate duplicate rows from the query answer. Include screenshots that show this.

Answer:

The following statement gives the first name of the agent table.

```
SELECT first
FROM agent
```

The screenshot shows the first six rows. There are totally 662 row(s). We can see the second and the sixth are the same.



first
Nick
Bill
Mathew
Jim
George
Bill

Question 12:

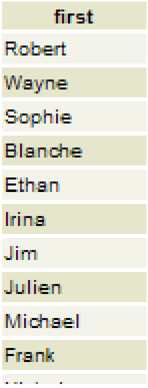
Write a similar query against the Spy database using the distinct clause that shows that the duplicate rows ARE eliminated.

Answer:

The following statement use “distinct” to eliminate duplicated rows.

```
SELECT DISTINCT first
FROM agent
```

The screenshot shows the first ten rows. There are totally 169 row(s) now. There are no duplicated rows.



first
Robert
Wayne
Sophie
Blanche
Ethan
Irina
Jim
Julien
Michael
Frank