

## Section One – Subqueries

### Step 1 – Creating Table Structure

```
Query Editor  Query History
171 INSERT INTO Sells(sells_id, store_location_id, product_id)
172 VALUES(1014, 13, 102);
173 INSERT INTO Sells(sells_id, store_location_id, product_id)
174 VALUES(1015, 13, 103);
175 INSERT INTO Sells(sells_id, store_location_id, product_id)
176 VALUES(1016, 13, 104);
177 INSERT INTO Offers(offers_id, store_location_id, shipping_offering_id)
178 VALUES(155, 13, 51);
179 INSERT INTO Offers(offers_id, store_location_id, shipping_offering_id)
180 VALUES(156, 13, 52);
181
182 --Toronto Extension
183 INSERT INTO Store_location(store_location_id, store_name, currency_accepted_id)
184 VALUES(14, 'Toronto Extension', 2);
185 INSERT INTO Sells(sells_id, store_location_id, product_id)
186 VALUES(1017, 14, 100);
187 INSERT INTO Sells(sells_id, store_location_id, product_id)
188 VALUES(1018, 14, 101);
189 INSERT INTO Sells(sells_id, store_location_id, product_id)
190 VALUES(1019, 14, 102);
191 INSERT INTO Sells(sells_id, store_location_id, product_id)
192 VALUES(1020, 14, 103);
193 INSERT INTO Sells(sells_id, store_location_id, product_id)
194 VALUES(1021, 14, 104);
195 INSERT INTO Offers(offers_id, store_location_id, shipping_offering_id)
196 VALUES(157, 14, 52);
```

Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 242 msec.

### Step 2 – Subquery in Column List

```
199 SELECT to_char(price_in_us_dollars *
200             (SELECT us_dollars_to_currency_ratio
201             FROM   Currency
202             WHERE  currency_name = 'British Pound'),
203             'FM£999D00') AS price_in_pounds
204 FROM   Product
205 WHERE  product_name = 'Digital Thermometer';
```

Data Output Explain Messages Notifications

	price_in_pounds text	
1	£167.50	

An uncorrelated subquery can be interpreted as the query containing its results independently from its outer statement, that it can be executed on its own outside of the outer query. Here, in

208	SELECT	us_dollars_to_currency_ratio
209	FROM	Currency
210	WHERE	currency_name = 'British Pound'
Data Output Explain Messages Notifications		
	us_dollars_to_currency_ratio	
	numeric (12,2)	
1		0.67

my query that if I only run: ; it works too. So this part is the uncorrelated subquery of my statement that gets the result of the price in British Pounds by multiplying with the US dollars currency ratio.

In this statement, this uncorrelated subquery helps retrieve the result as the ratio of the British pound is independent by itself, it would not change if the price of the product changes, and the ratio will be executed first so we get the correct price with its currency first. This uncorrelated subquery is solely on its own, and it will be executed first before the outer query.

### Step 3 – Subquery in WHERE Clause

a.

```

213 SELECT product_name,
214         to_char(price_in_us_dollars *
215                 (SELECT us_dollars_to_currency_ratio
216                   FROM Currency
217                   WHERE currency_name = 'Euro'),
218                 'FM€999D00') AS price_in_euros
219 FROM Product
220 WHERE price_in_us_dollars *
221        (SELECT us_dollars_to_currency_ratio
222          FROM Currency
223          WHERE currency_name = 'Euro') < 26
224 OR
225        price_in_us_dollars *
226        (SELECT us_dollars_to_currency_ratio
227          FROM Currency
228          WHERE currency_name = 'Euro') > 299;

```

	product_name	price_in_euros
	character varying (255)	text
1	Bag Valve Mask	€23.00
2	Electronic Stethoscope	€322.00
3	Handheld Pulse Oximeter	€414.00

b.

The SELECT selects the name of the product, and the formatted result in Euros based on the US dollar price multiplies to the US to Euro currency ratio, which the ratio value comes from the first subquery that independently serves as the us dollars to currency ratio from the Currency table where the currency name is Euro. The second subquery works in a similar way that converts the price to Euro but filters the price to less than 26. The third subquery works in a similar way that converts the price to Euro but filters the price to greater than 299. We then compare the product price from the product table and get the last result listed with the product\_name and price\_in\_euros as the columns we selected to present.

In short, the first subquery is used to retrieve the price in Euros, the second subquery is to restrict (filter) the products retrieved (price less than 26 Euros and more than 299 Euros).

#### Step 4 – Using the IN Clause with a Subquery

a.

```
SELECT Store_location.store_name,  
       Product.product_name,  
       Alternate_name.name AS alternate_name,  
       to_char(Product.price_in_us_dollars, 'FM$999.00') AS US_Price  
FROM   Store_location  
JOIN   Sells ON Sells.store_location_id = Store_location.store_location_id  
JOIN   Product ON Product.product_id = Sells.product_id  
JOIN   Alternate_name ON Alternate_name.product_id = Product.product_id  
WHERE  Product.product_id IN  
       (SELECT Product.product_id  
        FROM   Product  
        JOIN   Sells ON Sells.product_id = Product.product_id  
        GROUP BY Product.product_id  
        HAVING COUNT(Sells.product_id) = (SELECT COUNT(store_location_id) FROM Store_location))  
ORDER BY store_name, product_name, alternate_name;
```

--query

Data Output Explain Messages Notifications

	store_name character varying (255)	product_name character varying (255)	alternate_name character varying (255)	us_price text
1	Berlin Extension	Bag Valve Mask	Ambu Bag	\$25.00
2	Berlin Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
3	Berlin Extension	Digital Thermometer	Thermometer	\$250.00
4	Berlin Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
5	Berlin Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
6	Cancun Extension	Bag Valve Mask	Ambu Bag	\$25.00
7	Cancun Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
8	Cancun Extension	Digital Thermometer	Thermometer	\$250.00
9	Cancun Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
10	Cancun Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
11	London Extension	Bag Valve Mask	Ambu Bag	\$25.00
12	London Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
13	London Extension	Digital Thermometer	Thermometer	\$250.00
14	London Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
15	London Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
16	New York Extension	Bag Valve Mask	Ambu Bag	\$25.00
17	New York Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
18	New York Extension	Digital Thermometer	Thermometer	\$250.00
19	New York Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
20	New York Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
21	Toronto Extension	Bag Valve Mask	Ambu Bag	\$25.00
22	Toronto Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
23	Toronto Extension	Digital Thermometer	Thermometer	\$250.00
24	Toronto Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
25	Toronto Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00

--result

b.

Starting from the first part, I joined Store\_location to Sells, Product and Alternate\_Name, this outer query by itself would result in a list of store names (easier to see which location sells what) and all product names with corresponding alternative names (some products have different alternative names), and formatted product price in US dollars.

```

1 SELECT Store_location.store_name,
2        Product.product_name,
3        Alternate_name.name AS alternate_name,
4        to_char(Product.price_in_us_dollars, 'FM$999.00') AS US_Price
5 FROM Store_location
6 JOIN Sells ON Sells.store_location_id = Store_location.store_location_id
7 JOIN Product ON Product.product_id = Sells.product_id
8 JOIN Alternate_name ON Alternate_name.product_id = Product.product_id

```

Starting from the WHERE clause, I use this part of the subquery (highlighted below) to produce the list of all product\_id that exist in the Sells table.

```

WHERE Product.product_id IN
(SELECT Product.product_id
 FROM Product
 JOIN Sells ON Sells.product_id = Product.product_id
 GROUP BY Product.product_id
 HAVING COUNT(Sells.product_id) = (SELECT COUNT(store_location_id) FROM Store_location))
 ORDER BY store_name, product_name, alternate_name;

```

Then I am using the HAVING COUNT clause to limit the result to the product that is available in all store locations. The inner subquery will be executed first before the outer query.

Although there are total 5 store locations around the world at the current moment, we do not want to give a hard coded number since the number of the store may change as some close and some open, so I give it a dynamic number by using the subquery '(SELECT COUNT(store\_location\_id) FROM Store\_location)' to get an updated the number of the store location.

```

WHERE Product.product_id IN
(SELECT Product.product_id
 FROM Product
 JOIN Sells ON Sells.product_id = Product.product_id
 GROUP BY Product.product_id
 HAVING COUNT(Sells.product_id) = (SELECT COUNT(store_location_id) FROM Store_location))
 ORDER BY store_name, product_name, alternate_name;

```

Lastly, I used the WHERE clause with the IN operator, IN operator works to test whether the value is found in a list of values, so I can limit/filter the result to the products that are available in all store locations (the value from the inner query that has executed first) in this case.

```

WHERE Product.product_id IN
(SELECT Product.product_id
 FROM Product
 JOIN Sells ON Sells.product_id = Product.product_id
 GROUP BY Product.product_id
 HAVING COUNT(Sells.product_id) = (SELECT COUNT(store_location_id) FROM Store_location))
 ORDER BY store_name, product_name, alternate_name;

```

At the end, for virtual purposes, I used the ORDER BY keyword to sort the result in ascending order.

## Step 5 – Subquery in FROM Clause

a.

```
SELECT Store_location.store_name,
       Product.product_name,
       Alternate_name.name AS alternate_name,
       to_char(Product.price_in_us_dollars, 'FM$999.00') AS US_Price
FROM   (SELECT Product.product_id
        FROM   Product
        JOIN   Sells ON Sells.product_id = Product.product_id
        GROUP BY Product.product_id
        HAVING COUNT(Sells.product_id) = (SELECT COUNT(store_location_id) FROM Store_location)) information
JOIN   Sells ON Sells.product_id = information.product_id
JOIN   Store_location ON store_location.store_location_id = Sells.store_location_id
JOIN   Product ON Product.product_id = Sells.product_id
JOIN   Alternate_name ON Alternate_name.product_id = Product.product_id
ORDER BY store_name, product_name, alternate_name;
```

--query

	store_name character varying (255)	product_name character varying (255)	alternate_name character varying (255)	us_price text
1	Berlin Extension	Bag Valve Mask	Ambu Bag	\$25.00
2	Berlin Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
3	Berlin Extension	Digital Thermometer	Thermometer	\$250.00
4	Berlin Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
5	Berlin Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
6	Cancun Extension	Bag Valve Mask	Ambu Bag	\$25.00
7	Cancun Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
8	Cancun Extension	Digital Thermometer	Thermometer	\$250.00
9	Cancun Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
10	Cancun Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
11	London Extension	Bag Valve Mask	Ambu Bag	\$25.00
12	London Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
13	London Extension	Digital Thermometer	Thermometer	\$250.00
14	London Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
15	London Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
16	New York Extension	Bag Valve Mask	Ambu Bag	\$25.00
17	New York Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
18	New York Extension	Digital Thermometer	Thermometer	\$250.00
19	New York Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
20	New York Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
21	Toronto Extension	Bag Valve Mask	Ambu Bag	\$25.00
22	Toronto Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
23	Toronto Extension	Digital Thermometer	Thermometer	\$250.00
24	Toronto Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
25	Toronto Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00

--result

b.

We can successfully filter rows by using a subquery in the FROM clause in this query here. I change the subquery in WHERE clause in the Step #4 to the current FROM clause.

First, the subquery (highlighted in the screenshot below) executes and gets the result of the product\_id. This subquery is an uncorrelated subquery which can actually be returned in the outer query directly. Here, I named the result from this subquery 'information', this is an alias. This 'information' basically gets the result that produces the list of all product\_id that exist in the Sells table with a limitation that the product are available in all store locations.

1	SELECT	Store_location.store_name,
2		Product.product_name,
3		Alternate_name.name AS alternate_name,
4		to_char(Product.price_in_us_dollars, 'FM\$999.00') AS US_Price
5	FROM	(SELECT Product.product_id
6		FROM Product
7		JOIN Sells ON Sells.product_id = Product.product_id
8		GROUP BY Product.product_id
9		HAVING COUNT(Sells.product_id) = (SELECT COUNT(store_location_id) FROM Store_location)) information
10	JOIN	Sells ON Sells.product_id = information.product_id
11	JOIN	Store_location ON store_location.store_location_id = Sells.store_location_id
12	JOIN	Product ON Product.product_id = Sells.product_id
13	JOIN	Alternate_name ON Alternate_name.product_id = Product.product_id
14	ORDER BY	store_name, product_name, alternate_name;

Data Output		Explain	Messages	Notifications
	product_id			
	[PK] numeric (12)			
1	104			
2	102			
3	101			

Using a subquery in the FROM clause works as a table with the filtered result that we can join to the other tables. Here, I used the result obtained from the subquery which named 'information' to join the Sells table, and then I joined to Store\_location table to get the store\_name, Product table for the product name its price and Alternate\_name table for the corresponding alternative names (some product have different alternative names), and then I can get the list of the result I need for the store\_name, product\_name, alternate\_name and the us\_price. For virtual purposes, I used the ORDER BY keyword to sort the result in ascending order.



## Step 6 – Correlated Subquery

```

SELECT Store_location.store_name,
       Product.product_name,
       Alternate_name.name AS alternate_name,
       to_char(Product.price_in_us_dollars, 'FM$999.00') AS US_Price
FROM   Store_location
JOIN   Sells ON Sells.store_location_id = Store_location.store_location_id
JOIN   Product ON Product.product_id = Sells.product_id
JOIN   Alternate_name ON Alternate_name.product_id = Product.product_id
WHERE  EXISTS (SELECT universal.product_id FROM (
                SELECT product_id, COUNT(product_id) AS counting
                FROM   Sells
                GROUP BY Sells.product_id
                ) universal
          WHERE counting = (SELECT COUNT(store_location_id) FROM Store_location)
          AND universal.product_id = Product.product_id)
ORDER BY store_name, product_name, alternate_name

```

	store_name character varying (255)	product_name character varying (255)	alternate_name character varying (255)	us_price text
1	Berlin Extension	Bag Valve Mask	Ambu Bag	\$25.00
2	Berlin Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
3	Berlin Extension	Digital Thermometer	Thermometer	\$250.00
4	Berlin Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
5	Berlin Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
6	Cancun Extension	Bag Valve Mask	Ambu Bag	\$25.00
7	Cancun Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
8	Cancun Extension	Digital Thermometer	Thermometer	\$250.00
9	Cancun Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
10	Cancun Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
11	London Extension	Bag Valve Mask	Ambu Bag	\$25.00
12	London Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
13	London Extension	Digital Thermometer	Thermometer	\$250.00
14	London Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
15	London Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
16	New York Extension	Bag Valve Mask	Ambu Bag	\$25.00
17	New York Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
18	New York Extension	Digital Thermometer	Thermometer	\$250.00
19	New York Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
20	New York Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
21	Toronto Extension	Bag Valve Mask	Ambu Bag	\$25.00
22	Toronto Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
23	Toronto Extension	Digital Thermometer	Thermometer	\$250.00
24	Toronto Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
25	Toronto Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00

--result



- a. how your solution makes use of the correlated subquery and EXISTS clause to help retrieve the result

```
SELECT product_id, COUNT(product_id) AS counting
FROM Sells
GROUP BY Sells.product_id
```

I am using this subquery to return the counting times of a product in the Sells table, I am naming the result from the count as counting.

```
WHERE EXISTS (SELECT universal.product_id FROM (
    SELECT product_id, COUNT(product_id) AS counting
    FROM Sells
    GROUP BY Sells.product_id
) universal
WHERE counting = (SELECT COUNT(store_location_id) FROM Store_location)
AND universal.product_id = Product.product_id)
```

Then this subquery (highlighted)

filters the results and returns the product\_id that equals the count of the stores that exist (meaning the filtered products exist in all the stores that are currently open, the number can dynamically change). Then another condition in the WHERE clause works as a correlated

```
WHERE counting = (SELECT COUNT(store_location_id) FROM Store_location)
AND universal.product_id = Product.product_id)
```

subquery that correlates with the outer query. EXISTS returns true if there is at least a single row returned by the subquery in the parentheses, which works as to filter the outer values (in my case, all the products with its store name, alternative names and prices) and retrieve the relevant values that meets the criteria (in my case, the product\_id that exists in all store locations).

- b. how and when the correlated subquery is executed in the context of the outer query.

The WHERE clause inside the correlated subquery correlates the subquery with the outer query. A correlated subquery uses values from the outer query and inner query is executed when the statement is evaluated once for each row processed by the parent statement, and that meets the conditions of the WHERE clause. In my case, 'universal.product\_id = Product.product\_id' correlates the subquery to one that references the table introduced in the outer query.

## Step 7 – Using View in Query

```
1 CREATE OR REPLACE VIEW universal_products AS
2 SELECT Product.product_id
3 FROM Product
4 JOIN Sells ON Sells.product_id = Product.product_id
5 GROUP BY Product.product_id
6 HAVING COUNT(Sells.product_id) = (SELECT COUNT(store_location_id) FROM Store_location)
7
8 SELECT Store_location.store_name,
9 Product.product_name,
10 Alternate_name.name AS alternate_name,
11 to_char(Product.price_in_us_dollars, 'FM$999.00') AS US_Price
12 FROM universal_products
13 JOIN Sells ON Sells.product_id = universal_products.product_id
14 JOIN Store_location ON store_location.store_location_id = Sells.store_location_id
15 JOIN Product ON Product.product_id = Sells.product_id
16 JOIN Alternate_name ON Alternate_name.product_id = Product.product_id
17 ORDER BY store_name, product_name, alternate_name;
```

	store_name character varying (255)	product_name character varying (255)	alternate_name character varying (255)	us_price text
1	Berlin Extension	Bag Valve Mask	Ambu Bag	\$25.00
2	Berlin Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
3	Berlin Extension	Digital Thermometer	Thermometer	\$250.00
4	Berlin Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
5	Berlin Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
6	Cancun Extension	Bag Valve Mask	Ambu Bag	\$25.00
7	Cancun Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
8	Cancun Extension	Digital Thermometer	Thermometer	\$250.00
9	Cancun Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
10	Cancun Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
11	London Extension	Bag Valve Mask	Ambu Bag	\$25.00
12	London Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
13	London Extension	Digital Thermometer	Thermometer	\$250.00
14	London Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
15	London Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
16	New York Extension	Bag Valve Mask	Ambu Bag	\$25.00
17	New York Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
18	New York Extension	Digital Thermometer	Thermometer	\$250.00
19	New York Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
20	New York Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00
21	Toronto Extension	Bag Valve Mask	Ambu Bag	\$25.00
22	Toronto Extension	Bag Valve Mask	Oxygen Bag Valve Mask	\$25.00
23	Toronto Extension	Digital Thermometer	Thermometer	\$250.00
24	Toronto Extension	Handheld Pulse Oximeter	Handheld Pulse Oximeter Syst...	\$450.00
25	Toronto Extension	Handheld Pulse Oximeter	Portable Pulse Oximeter	\$450.00

--same result

## Section Two – Distributed Databases

### Step 8 – Simulating Horizontal Fragmentation

a. Create a table that has at least twelve rows and five columns. Make sure the table has a primary key.

	person_id [PK] numeric (12)	first_name character varying (32)	last_name character varying (32)	username character varying (20)	birthday date	four_digit_pin numeric (4)
1	1	Allan	Smith	allansmith123	1980-08-08	1234
2	2	Bryan	Brown	bryanbrown123	1981-08-08	5678
3	3	Charles	Williams	charleswilliams123	1982-08-08	2345
4	4	David	Johnson	davidjohnson123	1983-08-08	6789
5	5	Elan	Garcia	bryanbrown123	1984-08-08	3456
6	6	Frank	Davis	frankdavis123	1985-08-08	4567
7	7	Giorgio	Rodriguez	giorgiorodriguez123	1986-08-08	7890
8	8	Helen	Miller	helenmiller123	1994-08-08	1123
9	9	Ivy	Jones	ivyjones123	1988-08-08	2468
10	10	Jordon	Hernandez	jordonhernandez123	1991-08-08	1357
11	11	Kevin	Wilson	kevinwilson123	1993-08-08	3579
12	12	Lisa	Jackson	lisajackson123	1992-08-08	4680

```

1 CREATE TABLE Profile(
2   person_id DECIMAL(12) NOT NULL,
3   first_name VARCHAR(32) NOT NULL,
4   last_name VARCHAR(32) NOT NULL,
5   username VARCHAR(20) NOT NULL,
6   birthday DATE NOT NULL,
7   four_digit_pin DECIMAL(4) NOT NULL,
8   PRIMARY KEY (person_id));
9
10 CREATE SEQUENCE profile_seq START WITH 1;
11
12 INSERT INTO Profile
13 VALUES(nextval('profile_seq'), 'Allan', 'Smith', 'allansmith123', '08-08-1980', 1234);
14 INSERT INTO Profile
15 VALUES(nextval('profile_seq'), 'Bryan', 'Brown', 'bryanbrown123', '08-08-1981', 5678);
16 INSERT INTO Profile
17 VALUES(nextval('profile_seq'), 'Charles', 'Williams', 'charleswilliams123', '08-08-1982', 2345);
18 INSERT INTO Profile
19 VALUES(nextval('profile_seq'), 'David', 'Johnson', 'davidjohnson123', '08-08-1983', 6789);
20 INSERT INTO Profile
21 VALUES(nextval('profile_seq'), 'Elan', 'Garcia', 'bryanbrown123', '08-08-1984', 3456);
22 INSERT INTO Profile
23 VALUES(nextval('profile_seq'), 'Frank', 'Davis', 'frankdavis123', '08-08-1985', 4567);
24 INSERT INTO Profile
25 VALUES(nextval('profile_seq'), 'Giorgio', 'Rodriguez', 'giorgiorodriguez123', '08-08-1986', 7890);
26 INSERT INTO Profile
27 VALUES(nextval('profile_seq'), 'Helen', 'Miller', 'helenmiller123', '08-08-1994', 1123);
28 INSERT INTO Profile
29 VALUES(nextval('profile_seq'), 'Ivy', 'Jones', 'ivyjones123', '08-08-1988', 2468);
30 INSERT INTO Profile
31 VALUES(nextval('profile_seq'), 'Jordon', 'Hernandez', 'jordohnerandez123', '08-08-1991', 1357);
32 INSERT INTO Profile
33 VALUES(nextval('profile_seq'), 'Kevin', 'Wilson', 'kevinwilson123', '08-08-1993', 3579);
34 INSERT INTO Profile
35 VALUES(nextval('profile_seq'), 'Lisa', 'Jackson', 'lisajackson123', '08-08-1992', 4680);
36
37 SELECT * FROM Profile;

```

b. Define three views that simulate three horizontal fragments based upon some reasonable criteria. Show each view's contents.

```

47 CREATE OR REPLACE VIEW born_in_80s AS
48 SELECT *
49 FROM Profile
50 WHERE birthday < '01-01-1990';
51
52 SELECT * FROM born_in_80s;
53

```

Data Output Explain Messages Notifications

	person_id numeric (12)	first_name character varying (32)	last_name character varying (32)	username character varying (20)	birthday date	four_digit_pin numeric (4)
1	1	Allan	Smith	allansmith123	1980-08-08	1234
2	2	Bryan	Brown	bryanbrown123	1981-08-08	5678
3	3	Charles	Williams	charleswilliams123	1982-08-08	2345
4	4	David	Johnson	davidjohnson123	1983-08-08	6789
5	5	Elan	Garcia	bryanbrown123	1984-08-08	3456
6	6	Frank	Davis	frankdavis123	1985-08-08	4567
7	7	Giorgio	Rodriguez	giorgiorodriguez123	1986-08-08	7890
8	9	Ivy	Jones	ivyjones123	1988-08-08	2468

```

54 CREATE OR REPLACE VIEW born_in_90s AS
55 SELECT *
56 FROM Profile
57 WHERE birthday > '01-01-1990';
58
59 SELECT * FROM born_in_90s;

```

Data Output Explain Messages Notifications

	person_id numeric (12)	first_name character varying (32)	last_name character varying (32)	username character varying (20)	birthday date	four_digit_pin numeric (4)
1	8	Helen	Miller	helenmiller123	1994-08-08	1123
2	10	Jordon	Hernandez	jordonhernandez123	1991-08-08	1357
3	11	Kevin	Wilson	kevinwilson123	1993-08-08	3579
4	12	Lisa	Jackson	lisajackson123	1992-08-08	4680

```

61 CREATE OR REPLACE VIEW pin_0000_5000 AS
62 SELECT *
63 FROM Profile
64 WHERE four_digit_pin <= 5000;
65
66 SELECT * FROM pin_0000_5000;

```

Data Output Explain Messages Notifications

	person_id numeric (12)	first_name character varying (32)	last_name character varying (32)	username character varying (20)	birthday date	four_digit_pin numeric (4)
1	1	Allan	Smith	allansmith123	1980-08-08	1234
2	3	Charles	Williams	charleswilliams123	1982-08-08	2345
3	5	Elan	Garcia	bryanbrown123	1984-08-08	3456
4	6	Frank	Davis	frankdavis123	1985-08-08	4567
5	8	Helen	Miller	helenmiller123	1994-08-08	1123
6	9	Ivy	Jones	ivyjones123	1988-08-08	2468
7	10	Jordon	Hernandez	jordonhernandez123	1991-08-08	1357
8	11	Kevin	Wilson	kevinwilson123	1993-08-08	3579
9	12	Lisa	Jackson	lisajackson123	1992-08-08	4680

c. To simulate defragmentation, write and execute a query that combines the views to recreate the original table.

```

69 SELECT * FROM born_in_80s
70 UNION
71 SELECT * FROM born_in_90s
72 UNION
73 SELECT * FROM pin_0000_5000;

```

Data Output Explain Messages Notifications

	person_id numeric (12)	first_name character varying (32)	last_name character varying (32)	username character varying (20)	birthday date	four_digit_pin numeric (4)
1	10	Jordon	Hernandez	jordonhernandez123	1991-08-08	1357
2	9	Ivy	Jones	ivyjones123	1988-08-08	2468
3	5	Elan	Garcia	bryanbrown123	1984-08-08	3456
4	7	Giorgio	Rodriguez	giorgiorodriguez123	1986-08-08	7890
5	6	Frank	Davis	frankdavis123	1985-08-08	4567
6	8	Helen	Miller	helenmiller123	1994-08-08	1123
7	4	David	Johnson	davidjohnson123	1983-08-08	6789
8	3	Charles	Williams	charleswilliams123	1982-08-08	2345
9	1	Allan	Smith	allansmith123	1980-08-08	1234
10	2	Bryan	Brown	bryanbrown123	1981-08-08	5678
11	11	Kevin	Wilson	kevinwilson123	1993-08-08	3579
12	12	Lisa	Jackson	lisajackson123	1992-08-08	4680

## Step 9 – Simulating Vertical Fragmentation

a. Starting with the same logical table as in #8, define two views that simulate two vertical fragments based upon some reasonable column separation. Show each view's contents.

```
1 DROP VIEW name_birthday;
2 DROP VIEW username_pin;
3
4 CREATE OR REPLACE VIEW name_birthday AS
5 SELECT person_id, first_name, last_name, birthday
6 FROM Profile;
7
8 SELECT * FROM name_birthday;
9
```

Data Output Explain Messages Notifications

	person_id numeric (12)	first_name character varying (32)	last_name character varying (32)	birthday date
1	1	Allan	Smith	1980-08-08
2	2	Bryan	Brown	1981-08-08
3	3	Charles	Williams	1982-08-08
4	4	David	Johnson	1983-08-08
5	5	Elan	Garcia	1984-08-08
6	6	Frank	Davis	1985-08-08
7	7	Giorgio	Rodriguez	1986-08-08
8	8	Helen	Miller	1994-08-08
9	9	Ivy	Jones	1988-08-08
10	10	Jordon	Hernandez	1991-08-08
11	11	Kevin	Wilson	1993-08-08
12	12	Lisa	Jackson	1992-08-08

```
11 CREATE OR REPLACE VIEW username_pin AS
12 SELECT person_id, username, four_digit_pin
13 FROM Profile;
14
15 SELECT * FROM username_pin;
```

Data Output Explain Messages Notifications

	person_id numeric (12)	username character varying (20)	four_digit_pin numeric (4)
1	1	allansmith123	1234
2	2	bryanbrown123	5678
3	3	charleswilliams123	2345
4	4	davidjohnson123	6789
5	5	bryanbrown123	3456
6	6	frankdavis123	4567
7	7	giorgiorodriguez123	7890
8	8	helenmiller123	1123
9	9	ivyjones123	2468
10	10	jordonhernandez123	1357
11	11	kevinwilson123	3579
12	12	lisajackson123	4680

b. To simulate defragmentation, write and execute a query that combines the views to recreate the original table.

```
SELECT * FROM name_birthday
JOIN username_pin ON name_birthday.person_id = username_pin.person_id;
```

Data Output Explain Messages Notifications

	person_id numeric (12)	first_name character varying (32)	last_name character varying (32)	birthday date	person_id numeric (12)	username character varying (20)	four_digit_pin numeric (4)
1	1	Allan	Smith	1980-08-08	1	allansmith123	1234
2	2	Bryan	Brown	1981-08-08	2	bryanbrown123	5678
3	3	Charles	Williams	1982-08-08	3	charleswilliams123	2345
4	4	David	Johnson	1983-08-08	4	davidjohnson123	6789
5	5	Elan	Garcia	1984-08-08	5	bryanbrown123	3456
6	6	Frank	Davis	1985-08-08	6	frankdavis123	4567
7	7	Giorgio	Rodriguez	1986-08-08	7	giorgiorodriguez123	7890
8	8	Helen	Miller	1994-08-08	8	helenmiller123	1123
9	9	Ivy	Jones	1988-08-08	9	ivyjones123	2468
10	10	Jordon	Hernandez	1991-08-08	10	jordonhernandez123	1357
11	11	Kevin	Wilson	1993-08-08	11	kevinwilson123	3579
12	12	Lisa	Jackson	1992-08-08	12	lisajackson123	4680