Section One – Relating Data
Step 1 – Creating the Table Structure

CS669/postgres@PostgreSQL 13

Query Editor | Query History

```
1  CREATE TABLE Pizza (
2  pizza_id DECIMAL(12) PRIMARY KEY,
3  name VARCHAR(32) NOT NULL,
4  date_available DATE NOT NULL,
5  price DECIMAL(4, 2) NOT NULL);
6
7  CREATE TABLE Topping (
8  topping_id DECIMAL(12) PRIMARY KEY,
9  topping_name VARCHAR(64) NOT NULL,
10 pizza_id DECIMAL(12));
11
12 ALTER TABLE Topping
13 ADD CONSTRAINT topping_pizza_fk
14 FOREIGN KEY(pizza_id)
15 REFERENCES Pizza(pizza_id);
16
```

Data Output | Explain | Messages | Notifications

```
ALTER TABLE

Query returned successfully in 130 msec.
```

Step 2 – Populating the Tables

```
INSERT INTO Pizza (pizza_id, name, date_available, price)
VALUES (1, 'Plain', CAST('13-Jun-2020' AS DATE), 9.89);
INSERT INTO Pizza (pizza_id, name, date_available, price)
VALUES (2, 'Downtown Masterpiece', CAST('23-SEP-2020' AS DATE), 10.79);
INSERT INTO Pizza (pizza_id, name, date_available, price)
VALUES (3, 'Meat Lover', CAST('21-MAY-2021' AS DATE), 12.99);
INSERT INTO Pizza (pizza_id, name, date_available, price)
VALUES (4, 'Hawaiiaan', CAST('20-MAY-2021' AS DATE), 11.89);

INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (101, 'Tomatoes', 2);
INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (102, 'Spanich', 2);
INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (103, 'Pineapple', 4);
INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (104, 'Ham', 4);
INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (105, 'Onion', 3);
INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (106, 'Pepperoni', 3);
INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (107, 'Sausage', 3);
INSERT INTO Topping (topping_id, topping_name, pizza_id)
VALUES (108, 'Chicken', NULL);
```

SELECT * FROM Pizza;

```
43  SELECT * FROM Pizza;
```

Data Output | Explain | Messages | Notifications

	pizza_id [PK] numeric (12)	name character varying (32)	date_available date	price numeric (4,2)
1	1	Plain	2020-06-13	9.89
2	2	Downtown Masterpiece	2020-09-23	10.79
3	3	Meat Lover	2021-05-21	12.99
4	4	Hawaiiaan	2021-05-20	11.89

SELECT * FROM Topping;

```
45  SELECT * FROM Topping;
```

Data Output | Explain | Messages | Notifications

	topping_id [PK] numeric (12)	topping_name character varying (64)	pizza_id numeric (12)
1	101	Tomatoes	2
2	102	Spanich	2
3	103	Pineapple	4
4	104	Ham	4
5	105	Onion	3
6	106	Pepperoni	3
7	107	Sausage	3
8	108	Chicken	[null]

Step 3 – Invalid Reference Attempt

```
47  INSERT INTO Topping (topping_id, topping_name, pizza_id)
48  VALUES (109, 'Steak', 5);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  insert or update on table "topping" violates foreign key constraint "topping_pizza_fk"
DETAIL:  Key (pizza_id)=(5) is not present in table "pizza".
SQL state: 23503
```

- **Why did the insertion fail?**

The insertion failed because the database can not find the pizza_id with '5' in the Pizza table. Here. I tried to insert a row reference to the pizza_id that does not exist in the parent table. This action causes a foreign key violation - parent key not found. Constraint is a condition that has to be true all the time.

- **How you would interpret the error message from your RDBMS so that you know that the error indicates the Pizza reference is invalid.**

I am using Postgresql, the message shows as:
ERROR: insert or update on table "topping" violates foreign key constraint "topping_pizza_fk"
DETAIL: Key (pizza_id)=(5) is not present in table "pizza".
SQL state: 23503
The message I found to be pretty clear with the error I am making here, the pizza_id I tried to enter in the table 'Topping' violates foreign key constraint, because pizza_id=5 is not present in table "pizza". Here I set the foreign key constraint "topping_pizza_fk" I interpreted as the constraint defines a foreign key from Topping table to the Pizza table, the letters "fk" are an acronym to represent "foreign key"

Step 4 – Listing Pizzas and Toppings

```
50  SELECT name, topping_name
51  FROM Pizza
52  JOIN Topping on Topping.pizza_id = Pizza.pizza_id
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	topping_name character varying (64)
1	Downtown Masterpiece	Tomatoes
2	Downtown Masterpiece	Spanich
3	Hawaiiaan	Pineapple
4	Hawaiiaan	Ham
5	Meat Lover	Onion
6	Meat Lover	Pepperoni
7	Meat Lover	Sausage

- **Why some rows in the Pizza table and some rows in the Toppings table were not listed?**

This example is defined as an inner join, which contains only the things that match and then combines the tables together with only the matching records. Here I only list the names of the pizzas that have toppings, and the names of all of the toppings that go with each pizza. First, I selected name and topping_name as the two columns, then from Pizza table join to the Topping table. ON keyword indicates the Boolean (true/false) expression that will be used to determine which rows match. I do not see the 'Plain' pizza from the Pizza table and the 'Chicken' topping from the Topping table because both are exclusive to their own rows without any matching values from the condition Topping.pizza_id = Pizza.pizza_id.

Step 5 – Listing All Pizzas

LEFT JOIN

```
54  SELECT name, date_available, topping_name
55  FROM Pizza
56  LEFT JOIN Topping on Topping.pizza_id = Pizza.pizza_id
57  ORDER BY date_available;
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	date_available date	topping_name character varying (64)
1	Plain	2020-06-13	[null]
2	Downtown Masterpiece	2020-09-23	Spanich
3	Downtown Masterpiece	2020-09-23	Tomatoes
4	Hawaiiaan	2021-05-20	Ham
5	Hawaiiaan	2021-05-20	Pineapple
6	Meat Lover	2021-05-21	Onion
7	Meat Lover	2021-05-21	Pepperoni
8	Meat Lover	2021-05-21	Sausage

RIGHT JOIN

```
54  SELECT name, date_available, topping_name
55  FROM Topping
56  RIGHT JOIN Pizza on Topping.pizza_id = Pizza.pizza_id
57  ORDER BY date_available;
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	date_available date	topping_name character varying (64)
1	Plain	2020-06-13	[null]
2	Downtown Masterpiece	2020-09-23	Spanich
3	Downtown Masterpiece	2020-09-23	Tomatoes
4	Hawaiiaan	2021-05-20	Ham
5	Hawaiiaan	2021-05-20	Pineapple
6	Meat Lover	2021-05-21	Onion
7	Meat Lover	2021-05-21	Pepperoni
8	Meat Lover	2021-05-21	Sausage

Step 6 – Listing All Toppings

LEFT JOIN

```
59  SELECT topping_name, name
60  FROM Topping
61  Left JOIN Pizza on Topping.pizza_id = Pizza.pizza_id
62  ORDER BY topping_name DESC;
```

Data Output | Explain | Messages | Notifications

	topping_name character varying (64)	name character varying (32)
1	Tomatoes	Downtown Masterpiece
2	Spanich	Downtown Masterpiece
3	Sausage	Meat Lover
4	Pineapple	Hawaiiaan
5	Pepperoni	Meat Lover
6	Onion	Meat Lover
7	Ham	Hawaiiaan
8	Chicken	[null]

RIGHT JOIN

```
59  SELECT topping_name, name
60  FROM Pizza
61  RIGHT JOIN Topping on Topping.pizza_id = Pizza.pizza_id
62  ORDER BY topping_name DESC;
```

Data Output | Explain | Messages | Notifications

	topping_name character varying (64)	name character varying (32)
1	Tomatoes	Downtown Masterpiece
2	Spanich	Downtown Masterpiece
3	Sausage	Meat Lover
4	Pineapple	Hawaiiaan
5	Pepperoni	Meat Lover
6	Onion	Meat Lover
7	Ham	Hawaiiaan
8	Chicken	[null]

Step 7 – Listing All Pizzas and All Toppings

By pizza name

```
74  SELECT name, topping_name
75  FROM Pizza
76  FULL JOIN Topping on Topping.pizza_id = Pizza.pizza_id
77  ORDER BY name;
78
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	topping_name character varying (64)
1	Downtown Masterpiece	Tomatoes
2	Downtown Masterpiece	Spanich
3	Hawaiiaan	Pineapple
4	Hawaiiaan	Ham
5	Meat Lover	Onion
6	Meat Lover	Pepperoni
7	Meat Lover	Sausage
8	Plain	[null]
9	[null]	Chicken

By topping name

```
79  SELECT name, topping_name
80  FROM Pizza
81  FULL JOIN Topping on Topping.pizza_id = Pizza.pizza_id
82  ORDER BY topping_name;
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	topping_name character varying (64)
1	[null]	Chicken
2	Hawaiiaan	Ham
3	Meat Lover	Onion
4	Meat Lover	Pepperoni
5	Hawaiiaan	Pineapple
6	Meat Lover	Sausage
7	Downtown Masterpiece	Spanich
8	Downtown Masterpiece	Tomatoes
9	Plain	[null]

Section Two – Expressing Data

Step 8 – Formatting as Money

```
85  SELECT name, to_char(price, '$999.99') AS price
86  FROM Pizza;
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	price text
1	Plain	$ 9.89
2	Downtown Masterpiece	$ 10.79
3	Meat Lover	$ 12.99
4	Hawaiiaan	$ 11.89

Step 9 – Using Expressions

```
88  SELECT name, to_char(price - 1.75, '$999.99') AS discounted_price
89  FROM Pizza;
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	discounted_price text
1	Plain	$ 8.14
2	Downtown Masterpiece	$ 9.04
3	Meat Lover	$ 11.24
4	Hawaiiaan	$ 10.14

Step 10 – Advanced Formatting

```
94  SELECT topping_name || ' ( ' || name || ' - ' || to_char(price - 1.75, '$999.99') || ' ) ' AS discounted_price_advanced_formatting
95  FROM Topping
96  LEFT JOIN Pizza on Topping.pizza_id = Pizza.pizza_id
97  ORDER BY topping_name;
98
```

Data Output | Explain | Messages | Notifications

	discounted_price_advanced_formatting text
1	Ham (Hawaiiaan - $ 10.14)
2	Onion (Meat Lover - $ 11.24)
3	Pepperoni (Meat Lover - $ 11.24)
4	Pineapple (Hawaiiaan - $ 10.14)
5	Sausage (Meat Lover - $ 11.24)
6	Spanich (Downtown Masterpiece - $ 9.04)
7	Tomatoes (Downtown Masterpiece - $ 9.04)

Section Three – Advanced Data Expression

Step 11 – Evaluating Boolean Expressions

AND - the result turns out to be true if both operands are true. If any of the operands is false, the result is false.

OR - the result turns out to be true if any or both operands are true. The result turns false only if both conditions are false.

NOT - switch the result to be the opposite so that false becomes true and true becomes false.

a. (true AND false) OR (true AND true) = true

- true AND false = false
  Because for AND operators, if any of the operands is false, the result is false.
- true AND true = true
  Because for AND operators, the result turns out to be true if both operands are true.
- (true AND false) OR (true AND true) - false or true = true
  Because for OR operators, the result turns out to be true if any operand is true.

b. (true OR false) AND NOT(false OR false) AND (false AND true) = false

- true OR false = true
  Because for OR operators, the result turns out to be true if any operand is true.
- false OR false = false
  Because for OR operators, the result turns false only if both conditions are false.
- NOT(false OR false) = true
  Because for NOT operators, switch the result to be the opposite so that false becomes true.
- false AND true = false
  Because for AND operators, if any of the operands is false, the result is false.
- (true OR false) AND NOT(false OR false) AND (false AND true)
  true AND true AND false = false
  Because for AND operators, if any of the operands is false, the result is false.

c. NOT((false OR true) AND NOT(true AND true) AND (false OR true)) = true

- false OR true = true
  Because for OR operators, the result turns out to be true if any operand is true.
- true AND true = true
  Because for AND operators, the result turns out to be true if both operands are true.
- NOT(true AND true) = false
  Because for NOT operators, switch the result to be the opposite so that true becomes false.
- false OR true = true
  Because for OR operators, the result turns out to be true if any operand is true.
- (false OR true) AND NOT(true AND true) AND (false OR true)
  true AND false AND true = false
- NOT((false OR true) AND NOT(true AND true) AND (false OR true)) = true
  Because for NOT switch the result to be the opposite so that false becomes true.

Step 12 – Using Boolean Expressions in Queries

a. **signature pizza**

```
100  SELECT name, price
101  FROM Pizza
102  WHERE date_available >= CAST('01-MAY-2020' AS DATE)
103  AND price >= 9.55
104  AND NOT name = 'Plain'
105
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	price numeric (4,2)
1	Downtown Masterpiece	10.79
2	Meat Lover	12.99
3	Hawaiiaan	11.89

b. **flagship pizza**

```
107  INSERT INTO Pizza (pizza_id, name, date_available, price)
108  VALUES (5, 'Cheesy Lover', CAST('22-MAY-2021' AS DATE), 10);
109
110  SELECT name, price
111  FROM Pizza
112  WHERE date_available >= CAST('01-JAN-2021' AS DATE)
113  AND price <= 10
114
```

Data Output | Explain | Messages | Notifications

	name character varying (32)	price numeric (4,2)
1	Cheesy Lover	10.00

Step 13 – Using Generated Columns

a. Specials (half-price)

```
116  ALTER TABLE Pizza
117  ADD special_price decimal(4, 2) GENERATED ALWAYS AS (price / 2) STORED;
118
119  SELECT * FROM Pizza;
120
121
```

Data Output | Explain | Messages | Notifications

	pizza_id [PK] numeric (12)	name character varying (32)	date_available date	price numeric (4,2)	special_price numeric (4,2)
1	1	Plain	2020-06-13	9.89	4.95
2	2	Downtown Masterpiece	2020-09-23	10.79	5.40
3	3	Meat Lover	2021-05-21	12.99	6.50
4	4	Hawaiiaan	2021-05-20	11.89	5.95
5	5	Cheesy Lover	2021-05-22	10.00	5.00

b. Is_signature

```
122  ALTER TABLE Pizza
123  ADD is_signature BOOLEAN GENERATED ALWAYS AS (date_available >= CAST('01-MAY-2020' AS DATE)
124  AND price >= 9.55
125  AND NOT name = 'Plain')
126  STORED;
127
128  SELECT *
129  FROM Pizza
130  WHERE is_signature=true;
131
```

Data Output | Explain | Messages | Notifications

	pizza_id [PK] numeric (12)	name character varying (32)	date_available date	price numeric (4,2)	special_price numeric (4,2)	is_signature boolean
1	2	Downtown Masterpiece	2020-09-23	10.79	5.40	true
2	3	Meat Lover	2021-05-21	12.99	6.50	true
3	4	Hawaiiaan	2021-05-20	11.89	5.95	true
4	5	Cheesy Lover	2021-05-22	10.00	5.00	true