

# Jesse Opitz

## Database Management

### Lab 2

The image displays two screenshots of a PostgreSQL SQL Editor window, showing SQL queries and their results in the Output pane.

**Top Screenshot:**

The SQL Editor shows the following queries:

```
INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
VALUES(1026, 'may', 'c002', 'a05', 'p03', 800, 740.00);

-- SQL statements for displaying example data into the CAP3 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

select *
from customers;
--Notes from class v-----v please ignore
-- where city = 'Dallas';
-- where city != 'Dallas'
-- and discount < 9;

select *
from agents;

select *
from products;

select *
from orders;
```

The Output pane shows the results of the first query, displaying a table with 7 rows and 4 columns: **aid**, **name**, **city**, and **commission**.

aid	name	city	commission
1	a01	Smith New York	6.00
2	a02	Jones Newark	6.00
3	a03	Perry Tokyo	7.00
4	a04	Gray New York	6.00
5	a05	Otaai Duluth	5.00
6	a06	Smith Dallas	5.00
7	a08	Bond London	7.07

**Bottom Screenshot:**

The SQL Editor shows the same queries as the top screenshot, but the Output pane shows the results of the second query, displaying a table with 6 rows and 4 columns: **cid**, **name**, **city**, and **discount**.

cid	name	city	discount
1	c001	Tiptop Duluth	10.00
2	c002	Tyrell Dallas	12.00
3	c003	Allied Dallas	8.50
4	c004	ACME Duluth	8.00
5	c005	Weyland Acheron	0.00
6	c006	ACME Kyoto	0.00

The first screenshot shows a PostgreSQL query editor with the following SQL code:

```

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
VALUES(1026, 'may', 'c002', 'a05', 'p03', 800, 740.00);

-- SQL statements for displaying example data into the CAPS database
-- Connect to your Postgres server and set the active database to CAPS. Then . . .

select *
from customers;
--Notes from class v---v please ignore
-- where city = 'Dallas';
-- where city != 'Dallas'
-- and discount < 9;

select *
from agents;

select *
from products;

select *
from orders;

```

The output pane shows the following table:

	ordnum	mon	cid	aid	pid	qty	totalUSD
	integer	character(3)	character(4)	character(3)	character(3)	integer	numeric(12,2)
1	1011	jan	c001	a01	p01	1000	450.00
2	1013	jan	c002	a03	p03	1000	880.00
3	1015	jan	c003	a03	p05	1200	1104.00
4	1016	jan	c006	a01	p01	1000	500.00
5	1017	feb	c001	a06	p03	600	540.00
6	1018	feb	c001	a03	p04	600	540.00
7	1019	feb	c001	a02	p02	400	180.00
8	1020	feb	c006	a03	p07	600	600.00
9	1021	feb	c004	a06	p01	1000	440.00
10	1022	mar	c001	a05	p06	400	720.00
11	1023	mar	c001	a04	p05	500	450.00
12	1024	mar	c006	a06	p01	800	740.00

The second screenshot shows the same query editor with the following SQL code:

```

INSERT INTO Orders( ordnum, mon, cid, aid, pid, qty, totalUSD )
VALUES(1026, 'may', 'c002', 'a05', 'p03', 800, 740.00);

-- SQL statements for displaying example data into the CAPS database
-- Connect to your Postgres server and set the active database to CAPS. Then . . .

select *
from customers;
--Notes from class v---v please ignore
-- where city = 'Dallas';
-- where city != 'Dallas'
-- and discount < 9;

select *
from agents;

select *
from products;

select *
from orders;

```

The output pane shows the following table:

	pid	name	city	quantity	pricesUSD
	character(3)	text	text	integer	numeric(10,2)
1	p01	comb	Dallas	111400	0.50
2	p02	brush	Newark	203000	0.50
3	p03	razor	Duluth	190600	1.00
4	p04	pen	Duluth	125300	1.00
5	p05	pencil	Dallas	221400	1.00
6	p06	folder	Dallas	123100	2.00
7	p07	case	Newark	100500	1.00
8	p08	clip	Newark	200600	1.25

2) A primary key is the minimal number of columns to identify every column and is unique(EX: aid, cid, etc.). A candidate key is a superkey, with fewest possible amount of columns, still unique. Lastly, a superkey, is any field or any set of fields that uniquely identify every row in the table.

3) A data type is the type of data being entered into the table. Each field may have a different data type.

Cell Phones		
Category	Data type	Nullable/Not-nullable
cpid	INT PRIMARY KEY	Not-nullable
Name_of_Phone	CHAR(20)	Not-nullable
Height	INT	Not-nullable
Width	INT	Not-nullable
Creation_Date	DATE	Not-nullable
Phone_Brand	CHAR(20)	Not-nullable
Owner	CHAR(30)	Nullable

4) The “first normal form” rule is important because the intersection of rows and columns are atomic. “Access rows by content only” is important because it is the “what” of data. For example, you can request the row with 006 instead of where row 1 is. Lastly, the “all rows must be unique” rule makes it easier to query data because there won’t be any data duplication or confusion in the system. This is primarily used as a constant rule for the primary key.