

EXPLORING THE PARCH AND POSEY DATABASE (SQL ANALYSIS)

Data Analyst: Odudu Ochuko Richie

INTRODUCTION

In this analysis, I will be exploring the Parch and Posey database. I will carry out an analysis of this database to get insights that will be beneficial to the growth of the Parch and Posey company.

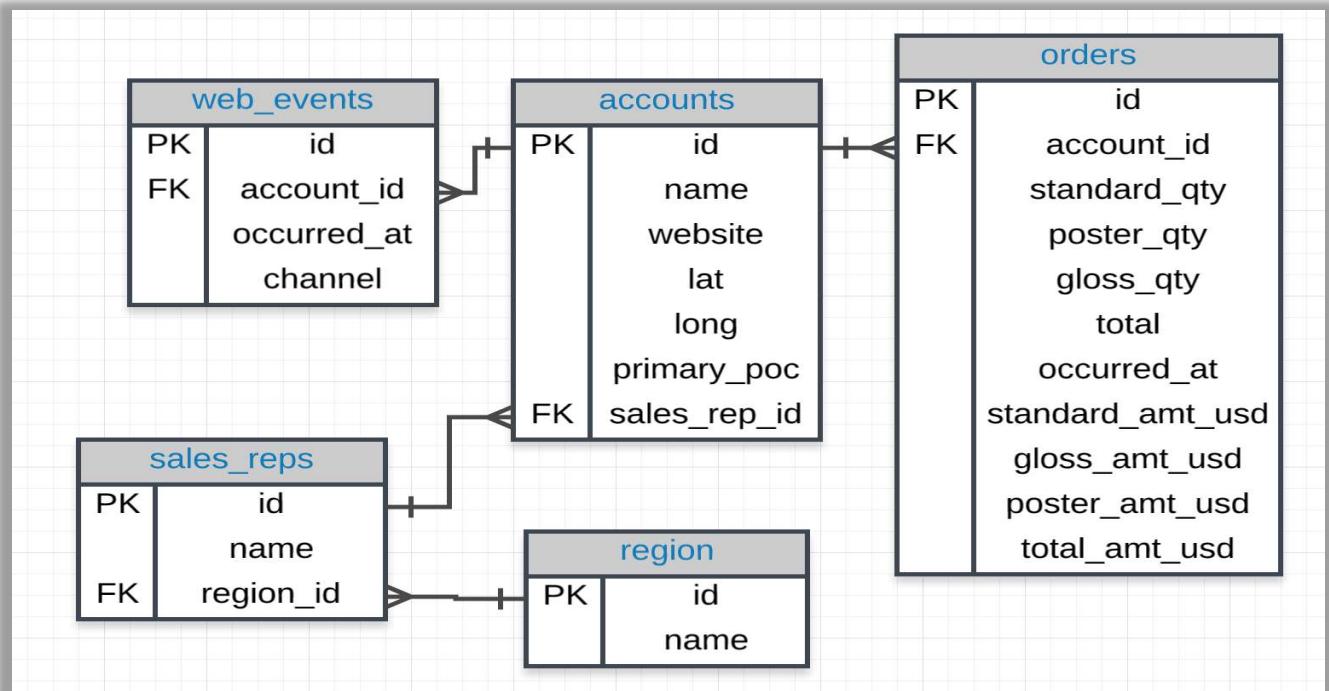
Firstly, Parch and Posey is a hypothetical company that sells paper. They have 50 sales reps spread across 4 regions in America. They sell 3 types of paper.

1. Standard
2. Gloss
3. Poster

Their customers are large companies they attract by advertising on Google, Facebook, and Twitter. In the Parch & Posey database, there are five tables web_events, accounts, orders, sales_reps, and region. The analysis will help understand the progress growth of Parch and Posey. In this analysis I will be looking at:

1. The account that placed the most orders
2. The account that placed the least orders
3. The account that placed the earliest order
4. The account that placed the latest order
5. Rank the total amount of paper ordered (from highest to lowest) for each account using a partition.
6. What kind of paper was ordered the most
7. Find the mean (average) amount spent per order on each paper type, as well as the mean amount of each paper type purchased per order.
8. The region for each sales rep along with their associated accounts. Sort the accounts alphabetically (A-Z) according to the account name.
9. Which year did Parch and Posey have the greatest sales in terms of total number of orders
10. Which month did Parch and Posey have the greatest sales in terms of total number of orders
11. The names of the sales reps in each region with the largest amount of sales (USD)
12. The region with the largest amount of total sales (USD), and how many total orders were placed?
13. The region with the smallest amount of total sales (USD), and how many total orders were placed?
14. The customer that spent the most (in total over their lifetime as a customer) in terms of total amount (USD).
How many web events did they have for each channel?
15. Provide the name of the sales rep in each region with the largest amount of total amount (USD).
16. The account that purchased the most (in total over their lifetime as a customer) in terms of standard quantity paper, how many accounts still had more in total?

THE PARCH AND POSEY ERD DIAGRAM



DATA CLEANING

1. Clean and restructure messy data
 2. Convert columns to different data types.
- A. In the **accounts** table, there is a column holding the **website** for each company. The last three digits specify what type of web address they are using. Pull these extensions and provide how many of each website type exist in the **accounts** table.

/* In the accounts table, there is a column holding the website for each company. The last three digits specify what type of web address they are using. Pull these extensions and provide how many of each website type exist in the accounts table. */

```

SELECT RIGHT(website, 3) AS domain, COUNT(*) num_companies
FROM accounts
GROUP BY 1
ORDER BY 2 DESC

```

domain	num_companies
com	349
org	1
net	1

Total rows: 3 of 3 Query complete 00:00:00.274 Ln 8, Col 16

B. There is much debate about how much the name (or even the first letter of a company name) matters.

Use the **accounts** table to pull the first letter of each company name to see the distribution of company names that begin with each letter (or number).

The screenshot shows the pgAdmin 4 interface with a query editor and a results table. The query is:`1 /*There is much debate about how much the name (or even the first letter of a company name) matters. Use the accounts
2 table to pull the first letter of each company name to see the distribution of company names
3 that begin with each letter (or number).*/
4
5 SELECT LEFT(UPPER(name), 1) AS first_letter, COUNT(*) num_companies
6 FROM accounts
7 GROUP BY 1
8 ORDER BY 2 DESC;`

The results table has two columns: first_letter (text) and num_companies (bigint). The data is:

first_letter	num_companies
A	37
C	37
P	27
M	22
D	17
T	17
S	17
B	16
L	16
E	16

Total rows: 26 of 26 Query complete 00:00:00.185 Ln 1, Col 3

C. Use the **accounts** table and a **CASE** statement to create two groups: one group of company names that start with a number and a second group of those company names that start with a letter. What proportion of company names start with a letter?

The screenshot shows the pgAdmin 4 interface with a query editor and a results table. The query is:`1 /* Use the accounts table and a CASE statement to create two groups: one group of company names that start with
2 a number and a second group of those company names that start with a letter.
3 What proportion of company names start with a letter? */
4
5 SELECT SUM(num) nums, SUM(letter) letters
6 FROM (SELECT name, CASE WHEN LEFT(UPPER(name), 1) IN ('0','1','2','3','4','5','6','7','8','9')
7 THEN 1 ELSE 0 END AS num,
8
9 CASE WHEN LEFT(UPPER(name), 1) IN ('0','1','2','3','4','5','6','7','8','9')
10 THEN 0 ELSE 1 END AS letter
11
12
13 FROM accounts) t1;`

The results table has two columns: nums (bigint) and letters (bigint). The data is:

nums	letters
1	350

Total rows: 1 of 1 Query complete 00:00:00.172 Ln 12, Col 16

D. Consider vowels as a, e, i, o, and u. What proportion of company names start with a vowel, and what percent start with anything else?

The screenshot shows the pgAdmin 4 interface with a database connection to 'parch_and_posey/postgres@PostgreSQL 15'. The left sidebar displays the schema browser with the 'Tables (5)' section selected, showing tables like accounts, orders, region, sales_reps, and web_events. The main query editor window contains the following SQL code:

```

1 /* Consider vowels as a, e, i, o, and u. What proportion of company names start with a vowel, and what percent
2 start with anything else? */
3
4 SELECT SUM(vowels) vowels, SUM(other) other
5 FROM (SELECT name, CASE WHEN LEFT(UPPER(name), 1) IN ('A','E','I','O','U')
6 THEN 1 ELSE 0 END AS vowels,
7 CASE WHEN LEFT(UPPER(name), 1) IN ('A','E','I','O','U')
8 THEN 0 ELSE 1 END AS other
9
10          FROM accounts) t1

```

The results are displayed in a Data Output tab, showing a single row with the values 80 and 271 for the 'vowels' and 'other' columns respectively.

E. Use the **accounts** table to create first and last name columns that hold the first and last names for the **primary_poc**.

The screenshot shows the pgAdmin 4 interface with the same database connection. The left sidebar shows the schema browser with the 'Tables (5)' section selected. The main query editor window contains the following SQL code:

```

1 /* Use the accounts table to create first and last name columns that hold
2 the first and last names for the primary_poc */
3
4 SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') - 1 ) first_name,
5      RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ') ) last_name
6
7 FROM accounts;

```

The results are displayed in a Data Output tab, showing 351 rows of data with two columns: 'first_name' and 'last_name'. The data includes names such as Tamara, Sung, Jodee, Serafina, Angeles, Savanna, Anabel, Barrie, Kym, Jamel, Parker, and Tuan.

- F. Now see if you can do the same thing for every rep **name** in the **sales_reps** table. Again, provide first and last name columns.

```

Dashboard Properties SQL Statistics Dependencies Dependents Processes
parch_and_posey/postgres@PostgreSQL 15*
Query History
Query
1 /* Now see if you can do the same thing for every rep name in the sales_reps table. Again provide
2 first and last name columns. */
3
4 SELECT LEFT(name, STRPOS(name, ' ')-1) AS first_name,
5      RIGHT(name, LENGTH(name)- STRPOS(name, ' ')) AS last_name
6 FROM sales_reps;
7
Data Output
Messages Notifications
first_name last_name
text       text
1 Samuel    Racine
2 Eugena   Esser
3 Michel   Averette
4 Renetta  Carew
5 Cara     Clarke
6 Lavera   Oles
7 Elba     Felder
8 Shawanda Selke
9 Sibyl    Lauria
10 Necole   Victory
11 Ernestine Pickron
12 Ananya   Morris
Total rows: 50 of 50 Query complete 00:00:00.207
Ln 6, Col 20

```

- G. Each company in the accounts table wants to create an email address for each primary_poc. The email address should be the first name of the **primary_poc**. last name **primary_poc** @ company name .com.

```

Dashboard Properties SQL Statistics Dependencies Dependents Processes
parch_and_posey/postgres@PostgreSQL 15*
Query History
Query
1 /* Each company in the accounts table wants to create an email address for each primary_poc. The email address
2 should be the first name of the primary_poc . last name primary_poc @ company name .com */
3
4 WITH t1 AS (
5     SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') -1 ) first_name,
6           RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name, name
7     FROM accounts)
8     SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@', name, '.com')
9   FROM t1;
10
Data Output
Messages Notifications
first_name last_name concat
text       text      text
1 Tamara    Tuma     Tamara.Tuma@Walmart.com
2 Sung      Shields  Sung.Shields@Exxon Mobil.com
3 Jodee     Lupo     Jodee.Lupo@Apple.com
4 Serafina  Banda   Serafina.Banda@Berkshire Hathaway.com
5 Angeles   Cruseo  Angeles.Crusoe@McKesson.com
6 Savanna   Gayman  Savanna.Gayman@UnitedHealth Group.com
7 Anabel    Haskell  Anabel.Haskell@CVS Health.com
8 Barrie    Omeara  Barrie.Omeara@General Motors.com
9 Kym       Hagerman Kym.Hagerman@Ford Motor.com
Total rows: 351 of 351 Query complete 00:00:00.185
Ln 10, Col 1

```

- H. You may have noticed that in the previous solution some of the company names include spaces, which will certainly not work in an email address. See if you can create an email address that will work by removing all of the spaces in the account name, but otherwise your solution should be just as in question 1

The screenshot shows the pgAdmin 4 interface. The left sidebar lists database objects like public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (5). The Tables section is expanded, showing accounts, orders, region, sales_reps, and web_events. The main window has a title bar for 'parch_and_posey/postgres@PostgreSQL 15*' and a toolbar with various icons. The 'Query' tab is selected, displaying a SQL script:

```

1 /* You may have noticed that in the previous solution some of the company names include spaces, which will certainly
2 not work in an email address. See if you can create an email address that will work by
3 removing all of the spaces in the account name, but otherwise your solution should be just as in question 1. */
4
5 WITH t1 AS (
6     SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') - 1) first_name,
7         RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name, name
8     FROM accounts
9     SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@', REPLACE(name, ' ', ''), '.com')
10    FROM t1;
11

```

The 'Data Output' tab shows a table with columns: first_name, last_name, and concat. The data is:

	first_name	last_name	concat
1	Tamara	Tuma	Tamara.Tuma@Walmart.com
2	Sung	Shields	Sung.Shields@ExxonMobil.com
3	Jodee	Lupo	Jodee.Lupo@Apple.com
4	Serafina	Banda	Serafina.Banda@BerkshireHathaway.com
5	Angeles	Crusoe	Angeles.Crusoe@McKesson.com
6	Savanna	Gayman	Savanna.Gayman@UnitedHealthGroup.com
7	Anabel	Haskell	Anabel.Haskell@CVSHealth.com
8	Barrie	Omeara	Barrie.Omeara@GeneralMotors.com

Total rows: 351 of 351 Query complete 00:00:00.201 Ln 11, Col 1

- I. We would also like to create an initial password, which they will change after their first log in. The first password will be the first letter of the primary_poc's first name (lowercase), then the last letter of their first name (lowercase), the first letter of their last name (lowercase), the last letter of their last name (lowercase), the number of letters in their first name, the number of letters in their last name, and then the name of the company they are working with, all capitalized with no spaces

The screenshot shows the pgAdmin 4 interface. The left sidebar lists database objects like public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (5). The Tables section is expanded, showing accounts, orders, region, sales_reps, and web_events. The main window has a title bar for 'parch_and_posey/postgres@PostgreSQL 15*' and a toolbar with various icons. The 'Query' tab is selected, displaying a SQL script:

```

1 /* We would also like to create an initial password, which they will change after their first log in.
2 The first password will be the first letter of the primary_poc's first name (lowercase), then the last letter of their
3 first name (lowercase), the first letter of their last name (lowercase), the last letter of their last name (lowercase),
4 the number of letters in their first name, the number of letters in their last name, and then the name of the company
5 they are working with, all capitalized with no spaces.
6 */
7 WITH t1 AS (
8     SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') - 1) first_name,
9         RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name, name
10    FROM accounts
11     SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@', name, '.com'),
12           LEFT(LOWER(first_name), 1) || RIGHT(LOWER(first_name), 1) || LEFT(LOWER(last_name), 1),
13           || RIGHT(LOWER(last_name), 1) || LENGTH(first_name) || LENGTH(last_name) || REPLACE(UPPER(name), ' ', '')
14    FROM t1;
15

```

The 'Data Output' tab shows a table with columns: first_name, last_name, concat, and ?column? (which is highlighted in a blue box). The data is:

	first_name	last_name	concat	?column?
1	Tamara	Tuma	Tamara.Tuma@Walmart.com	tata64WALMART
2	Sung	Shields	Sung.Shields@Exxon Mobil.com	sgs47EXXON MOBIL
3	Jodee	Lupo	Jodee.Lupo@Apple.com	jelo54APPLE
4	Serafina	Banda	Serafina.Banda@Berkshire Hathaway.com	saba85BERKSHIRE HATHAWAY

Total rows: 351 of 351 Query complete 00:00:00.227 Ln 13, Col 113

DATA ANALYSIS

1. The Account that Placed the Most Orders:

The screenshot shows the pgAdmin interface with a database connection to 'parch_and_posey/postgres@PostgreSQL 15'. The left sidebar shows the schema structure with 'Tables (5)' selected, containing 'accounts', 'orders', 'region', 'sales_reps', and 'web_events'. The main window displays a SQL query and its results. The query is:

```
1 /* I want to find the account that placed the most orders */
2
3 SELECT a.id, a.name, COUNT(*) num_orders
4 FROM accounts a
5 JOIN orders o
6 ON a.id = o.account_id
7 GROUP BY a.id, a.name
8 ORDER BY num_orders DESC
9 LIMIT 1
```

The results table shows one row:

	id	name	num_orders
1	3411	Leucadia National	71

Total rows: 1 of 1 Query complete 00:00:00.180 Ln 1, Col 30

"Leucadia National" placed the most orders. This account placed 71 orders in total.

2. The Account that Placed the Least Orders:

The screenshot shows the pgAdmin interface with the same database connection and schema. The main window displays a SQL query and its results. The query is identical to the one above:

```
1 /* I want to find the account that placed the least orders */
2
3 SELECT a.id, a.name, COUNT(*) num_orders
4 FROM accounts a
5 JOIN orders o
6 ON a.id = o.account_id
7 GROUP BY a.id, a.name
8 ORDER BY num_orders DESC
9 LIMIT 1
```

The results table shows one row:

	id	name	num_orders
1	1591	Lockheed Martin	1

Total rows: 1 of 1 Query complete 00:00:00.144 Ln 1, Col 30

"Lockheed Martin" placed the least orders. This account placed 1 order in total.

3. The Account that Placed the Earliest Order:

The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects under the schema 'public'. The 'Tables' node is expanded, showing five tables: accounts, orders, region, and sales_reps. The 'orders' table is selected. The main window contains a query editor with the following SQL code:

```
1 /* Finding the account that placed the earliest order */
2
3 SELECT a.name, o.occurred_at
4 FROM accounts a
5 JOIN orders o
6 ON a.id = o.account_id
7 ORDER BY o.occurred_at
8 LIMIT 1
```

Below the query editor is a 'Data Output' tab showing the result of the query:

	name	occurred_at
1	DISH Network	2013-12-04 04:22:44

At the bottom of the interface, status bars show 'Total rows: 1 of 1' and 'Query complete 00:00:00.114'.

"DISH Network" placed the earliest order in "2013-12-04 04:22:44"

4. The Account that Placed the Latest Order:

The screenshot shows the pgAdmin 4 interface, similar to the previous one but with a different query. The tree view on the left shows the same schema structure. The 'Tables' node is expanded, and the 'orders' table is selected. The main window contains a query editor with the following SQL code:

```
1 /* Finding the account that placed the latest order */
2
3 SELECT a.name, o.occurred_at
4 FROM accounts a
5 JOIN orders o
6 ON a.id = o.account_id
7 ORDER BY o.occurred_at DESC
8 LIMIT 1
```

Below the query editor is a 'Data Output' tab showing the result of the query:

	name	occurred_at
1	W.W. Grainger	2017-01-02 00:02:40

At the bottom of the interface, status bars show 'Total rows: 1 of 1' and 'Query complete 00:00:00.403'.

"W.W. Grainger" placed the latest order in "2017-01-02 00:02:40"

5. Rank the Total Amount of Paper Ordered (From Highest to Lowest) for Each Account using a Partition:

The screenshot shows the pgAdmin interface with a query editor and a results table.

Query Editor:

```

1 /* Ranking the total amount of paper ordered (from highest to lowest) for each account using a partition */
2
3 SELECT id, account_id, total,
4        RANK() OVER (PARTITION BY account_id ORDER BY total DESC) AS total_rank
5 FROM orders
6

```

Data Output:

	id	account_id	total	total_rank
1	4308	1001	1410	1
2	4309	1001	1405	2
3	4316	1001	1384	3
4	4317	1001	1347	4
5	4314	1001	1343	5
6	4307	1001	1321	6
7	4311	1001	1307	7
8	4310	1001	1280	8
9	4312	1001	1267	9
10	4313	1001	1254	10
11	4315	1001	1238	11

Total rows: 1000 of 6912 | Query complete 00:00:00.346 | Ln 6, Col 1

6. What Kind of Paper was Ordered the Most:

The screenshot shows the pgAdmin interface with a query editor and a results table.

Query Editor:

```

1 /* What kind of paper was ordered the most */
2
3 SELECT SUM(standard_qty) standard_paper, SUM(poster_qty) poster_paper, SUM(gloss_qty) gloss_paper
4 FROM orders
5 ORDER BY 1, 2, 3 DESC

```

Data Output:

	standard_paper	poster_paper	gloss_paper
1	1938346	723646	1013773

Total rows: 1 of 1 | Query complete 00:00:00.511 | Ln 1, Col 1

Standard paper had **1938346** orders, Post paper had **723646** orders, Gloss paper had **1013773** orders. This analysis showed that **Standard paper** had the highest orders which was one million nine hundred thirty-eight thousand three hundred forty-six.

7. Find the Mean (Average) Amount Spent Per Order on Each Paper Type, as well as the Mean Amount of Each Paper type Purchased Per Order:

The screenshot shows the pgAdmin 4 interface with a database connection to 'parch_and_posey/postgres@PostgreSQL 15'. The left sidebar displays the schema structure with tables like accounts, orders, region, sales_reps, and web_events. The main area contains a query editor with the following SQL code:

```

1 /*Find the mean (average) amount spent per order on each paper type, as well as the mean amount of each paper
2 type purchased per order. */
3
4 SELECT AVG(standard_qty) AS avg_standar_qty,
5      AVG(gloss_qty) AS avg_gloss_qty,
6      AVG(poster_qty) AS avg_poster_qty,
7      AVG(standard_amt_usd) AS avg_standard_amt,
8      AVG(gloss_amt_usd) AS avg_gloss_amt,
9      AVG(poster_amt_usd) AS avg_poster_amt
10 FROM orders;
    
```

The results are displayed in a data grid:

	avg_standar_qty	avg_gloss_qty	avg_poster_qty	avg_standard_amt	avg_gloss_amt	avg_poster_amt
1	280.432002314814818	146.6685474537037037	104.6941550925925925926	1399.3556915509259259259	1098.5474204282407407	850.1165393518518519

Total rows: 1 of 1 Query complete 00:00:00.198 Ln 10, Col 13

8. The Region for Each Sales Rep Along with their Associated Accounts. Sort the Accounts Alphabetically (A-Z) According to the Account Name:

The screenshot shows the pgAdmin 4 interface with a database connection to 'parch_and_posey/postgres@PostgreSQL 15'. The left sidebar displays the schema structure with tables like accounts, orders, region, sales_reps, and web_events. The main area contains a query editor with the following SQL code:

```

1 /* The region for each sales rep along with their associated accounts. Sort the accounts alphabetically (A-Z)
2 according to the account name.*/
3 SELECT r.name region, s.name rep, a.name account
4 FROM sales_reps s
5 JOIN region r
6 ON s.region_id = r.id
7 JOIN accounts a
8 ON a.sales_rep_id = s.id
9 ORDER BY a.name
10 LIMIT 10;
    
```

The results are displayed in a data grid:

	region	rep	account
1	Northeast	Sibyl Lauria	3M
2	Midwest	Chau Rowles	Abbott Laboratories
3	Midwest	Julie Starr	AbbVie
4	Southeast	Earlie Schleusner	ADP
5	West	Marquetta Laycock	Advance Auto Parts
6	Southeast	Moon Torian	AECOM
7	Southeast	Calvin Ollison	AES
8	Northeast	Renetta Carew	Aetna

Total rows: 10 of 10 Query complete 00:00:00.133 4 new notifications

9. Which Year Did Parch and Posey have the Greatest Sales in Terms of Total Number of Orders:

The screenshot shows the pgAdmin 4 interface with a database connection named 'parch_and_posey/postgres@PostgreSQL 15'. The left sidebar displays various database objects like public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (5). The 'Tables (5)' section is selected, showing tables: accounts, orders, region, sales_reps, and web_events. The main query editor window contains the following SQL code:

```
1 /* Which year did Parch and Posey have the greatest sales in terms of total number of orders */
2
3 SELECT DATE_PART('year', occurred_at) ord_year,
4        COUNT(*) total_sales
5 FROM orders
6 GROUP BY 1
7 ORDER BY 2 DESC;
```

The results are displayed in a table titled 'Data Output' with columns 'ord_year' (double precision) and 'total_sales' (bigint). The data is as follows:

ord_year	total_sales
2016	3757
2015	1725
2014	1306
2013	99
2017	25

Total rows: 5 of 5 Query complete 00:00:00.542 Ln 7, Col 17

Parch and Posey had their greatest sales in **2016** with over **3757 total sales**.

10. Which Month did Parch and Posey have the Greatest Sales in Terms of Total Number of Orders:

The screenshot shows the pgAdmin 4 interface with the same database connection and sidebar as the previous screenshot. The main query editor window contains the same SQL code as question 9:

```
1 /* Which year did Parch and Posey have the greatest sales in terms of total number of orders */
2
3 SELECT DATE_PART('year', occurred_at) ord_year,
4        COUNT(*) total_sales
5 FROM orders
6 GROUP BY 1
7 ORDER BY 2 DESC;
```

The results are displayed in a table titled 'Data Output' with columns 'ord_year' (double precision) and 'total_sales' (bigint). The data is identical to the previous screenshot:

ord_year	total_sales
2016	3757
2015	1725
2014	1306
2013	99
2017	25

Total rows: 5 of 5 Query complete 00:00:00.542 Ln 7, Col 17

Parch and Posey had their greatest sales in the month of **December** with over **882 total sales**

11. The Names of the Sales Reps in each Region with the Largest Amount of Sales (USD):

The screenshot shows the pgAdmin interface with a query editor window. The query is:`1 /* The name of the sales rep in each region with the largest amount of sales (USD) */
2
3 SELECT s.name rep, r.name region, SUM(o.total_amt_usd) total_sales
4 FROM region r
5 JOIN sales_reps s
6 ON r.id = s.region_id
7 JOIN accounts a
8 ON s.id = a.sales_rep_id
9 JOIN orders o
10 ON a.id = o.account_id
11 GROUP BY s.name, r.name
12 ORDER BY total_sales DESC
13 LIMIT 5;`

The results table shows five rows:

	rep	region	total_sales
1	Earlie Schleusner	Southeast	1098137.72
2	Tia Amato	Northeast	1010690.60
3	Vernita Plump	Southeast	934212.93
4	Georgianna Chisholm	West	886244.12
5	Arica Stoltzfus	West	810353.34

Total rows: 5 of 5 Query complete 00:00:00.173 Ln 9, Col 14

"Earlie Schleusner" from the "Southeast" region had the largest amount of sales (**1098137.72**)

12. The Region with the Largest Amount of Total Sales (USD), and How Many Total Orders were Placed:

The screenshot shows the pgAdmin interface with a query editor window. The query is:`1 /* The region with the largest amount of total sales (USD), and how many total orders were placed? */
2
3 SELECT r.name region, SUM(o.total_amt_usd) largest_sales_usd,
4 COUNT(o.total) total_orders
5 FROM region r
6 JOIN sales_reps s
7 ON r.id = s.region_id
8 JOIN accounts a
9 ON s.id = a.sales_rep_id
10 JOIN orders o
11 ON a.id = o.account_id
12 GROUP BY 1
13 ORDER BY 2 DESC, 3;`

The results table shows four rows:

region	largest_sales_usd	total_orders
Northeast	7744405.36	2357
Southeast	6458497.00	2024
West	5925122.96	1634
Midwest	3013486.51	897

Total rows: 4 of 4 Query complete 00:00:00.200 Ln 13, Col 20

"Northeast" had the largest amount of total sales (**7744405.36**) in USD and **2357** orders were placed from that region.

13. The Region with the Smallest Amount of Total Sales (USD), and How Many Total Orders were Placed:

The screenshot shows the pgAdmin interface with a database connection named 'parch_and_posey/postgres@PostgreSQL 15'. The left sidebar displays the schema structure, including tables like accounts, orders, region, sales_reps, and web_events. The main window contains a SQL query and its results.

```
/* The region with the smallest amount of total sales (USD), and how many total orders were placed? */
SELECT r.name region, SUM(o.total_amt_usd) smallest_sales_usd,
       COUNT(o.total) total_orders
FROM region r
JOIN sales_reps s
ON r.id = s.region_id
JOIN accounts a
ON s.id = a.sales_rep_id
JOIN orders o
ON a.id = o.account_id
GROUP BY 1
ORDER BY 2 ASC, 3;
```

	region	smallest_sales_usd	total_orders
1	Midwest	3013486.51	897
2	West	5925122.96	1634
3	Southeast	6458497.00	2024
4	Northeast	7744405.36	2357

Total rows: 4 of 4 Query complete 00:00:00.438 Ln 10, Col 14

"Midwest" had the smallest amount of total sales (**3013486.51**) in USD and **897** orders were placed from that region

14. The Customer that Spent the Most (In Total Over their Lifetime as a Customer) in Terms of Total Amount (USD). How Many Web Events did they Have for Each Channel:

The screenshot shows the pgAdmin interface with a database connection named 'parch_and_posey/postgres@PostgreSQL 15'. The left sidebar displays the schema structure, including tables like accounts, orders, region, sales_reps, and web_events. The main window contains a SQL query and its results.

```
/* The customer that spent the most (in total over their lifetime as a customer) in terms of total amount (USD).
How many web events did they have for each channel? */

SELECT a.name, w.channel, COUNT(*) num_events
FROM accounts a
JOIN web_events w
ON a.id = w.account_id AND a.id = (SELECT id
                                     FROM(SELECT a.id, a.name, SUM(o.total_amt_usd) total_spent
                                         FROM orders o
                                         JOIN accounts a
                                         ON a.id = o.account_id
                                         GROUP BY a.id, a.name
                                         ORDER BY 3 DESC
                                         LIMIT 1) inner_table)
GROUP BY 1,2
ORDER BY 3 DESC;
```

	channel	num_events
1	Search	1000
2	Referrals	800
3	Direct	600
4	Social	400
5	Email	200

Total rows: 6 of 6 Query complete 00:00:00.175 Ln 4, Col 46

Dashboard Properties SQL Statistics Dependencies Dependents Processes 14.The customer that spent the most (in total over their life) < > x

public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
accounts
orders
region
sales_reps
web_events
Trigger Functions
Types
Views
Subscriptions
postgres
n/Group Roles
espaces

Query Query History

```
1 /* The customer that spent the most (in total over their lifetime as a customer) in terms of total amount (USD).
2 How many web events did they have for each channel? */
3
```

Data Output Messages Notifications

	name character	channel character	num_events bigint
1	EOG Resources	direct	44
2	EOG Resources	organic	13
3	EOG Resources	adwords	12
4	EOG Resources	facebook	11
5	EOG Resources	twitter	5
6	EOG Resources	banner	4

Total rows: 6 of 6 Query complete 00:00:00.235 Ln 17, Col 1

EOG Resources spent the most in total over their lifetime as a customer terms of total amount in us dollars, thy had 89 number of web events all together.

15. Provide the Name of the Sales Rep in Each Region with the Largest Amount of Total Amount (USD):

Accesses 15.Provide the name of the sales rep in each region with the largest amount of total amount (USD)..sql parch_and_pos... parch_and_pos...

public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
accounts
orders
region
sales_reps
web_events
Trigger Functions
Types
Views
Subscriptions
postgres
n/Group Roles
espaces

Query Query History

```
1 /* Provide the name of the sales rep in each region with the largest amount of total amount (USD) */
2
3 WITH t1 AS(
4     SELECT s.name rep_name, r.name region, SUM(o.total_amt_usd) total_amt
5     FROM sales_reps s
6     JOIN accounts a
7     ON a.sales_rep_id = s.id
8     JOIN orders o
9     ON o.account_id = a.id
10    JOIN region r
11    ON r.id = s.region_id
12    GROUP BY 1,2
13    ORDER BY 3 DESC),
14 t2 AS(
15     SELECT region, MAX(total_amt) total_amt
16     FROM t1
17     GROUP BY 1)
18 SELECT t1.rep_name, t1.region, t1.total_amt
19 FROM t1
20 JOIN t2
21 ON t1.region = t2.region AND t1.total_amt = t2.total_amt;
22
```

Total rows: 4 of 4 Query complete 00:00:00.118 Ln 21, Col 58

15. Provide the name of the sales rep in each region with the largest amount of total amount (USD)..sql

```

1 /* Provide the name of the sales rep in each region with the largest amount of total amount (USD) */
2

```

	rep_name	region	total_amt
1	Earlie Schleusner	Southeast	1098137.72
2	Tia Amato	Northeast	1010690.60
3	Georgianna Chisholm	West	886244.12
4	Charles Bidwell	Midwest	675637.19

Total rows: 4 of 4 Query complete 00:00:00.130 Ln 21, Col 58

16. The Account that Purchased the Most (In Total Over their Lifetime as a Customer) in Terms of Standard Quantity Paper, How Many Accounts still had More in Total:

16. The account that purchased the most (in total over their lifetime as a customer) in terms of standard quantity paper, how many accounts still had more in total?

```

1 /* The account that purchased the most (in total over their lifetime as a customer) in terms of standard quantity
2 paper, how many accounts still had more in total */
3 WITH t1 AS(
4     SELECT a.name account_name, SUM(o.standard_qty)total_std, SUM(o.total) total
5     FROM accounts a
6     JOIN orders o
7     ON o.account_id = a.id
8     GROUP BY 1
9     ORDER BY 2 DESC
10    LIMIT 1),
11 t2 AS(
12     SELECT a.name name
13     FROM orders o
14     JOIN accounts a
15     ON a.id = o.account_id
16     GROUP BY 1
17     HAVING SUM (o.total) > (SELECT total FROM t1)
18     SELECT COUNT(*) accounts_had_more_in_total
19     FROM t2

```

	accounts_had_more_in_total
1	3

Total rows: 1 of 1 Query complete 00:00:00.113 Ln 11, Col 8

In terms of standard paper, 3 accounts still had more in total.

CONCLUSIONS

- In my conclusion, I found out that **Leucadia National**" placed the most orders, and this account placed 71 orders in total.
- I also found that "**Lockheed Martin**" placed the least orders, and this account placed 1 order in total.
- **DISH Network** placed the earliest order in **2013-12-04 04:22:44** and **W.W. Grainger**" placed the latest order in **2017-01-02 00:02:40**.
- **Standard paper** had **1938346** orders, **Post paper** had **723646** orders, **Gloss paper** had **1013773** orders. This analysis showed that **Standard paper** had the highest orders which was one million nine hundred thirty-eight thousand three hundred forty-six.
- I found out that Parch and Posey had their greatest sales in **2016** with over **3757 total sales** and precisely had their greatest sales in the month of **December** with over **882 total sales**
- **Earlie Schleusner** from the "**Southeast**" region had the largest amount of sales (**1098137.72**)
- **Northeast** had the largest amount of total sales (**7744405.36**) in USD and **2357** orders were placed from that region.
- **Midwest** had the smallest amount of total sales (**3013486.51**) in USD and **897** orders were placed from that region
- **EOG Resources** spent the most in total over their lifetime as a customer in terms of total amount in US dollars, they had 89 number of web events all together.
- In terms of standard paper, 3 accounts still had more in total.