Introduction : Sales Analysis

Sales analysis is a crucial process that involves examining and interpreting sales data to gain valuable insights into the performance of a business. By analyzing sales data, companies can make informed decisions, identify trends, and develop effective strategies to drive growth and improve profitability. In the context of the provided PostgreSQL database, we will explore the sales data related to paper orders from various companies to uncover meaningful information.

Goal of the Project:

The goal of this project is to conduct a comprehensive sales analysis using the PostgreSQL database to uncover valuable insights about the paper orders placed by various companies. The specific objectives are to identify the best-selling products, determine the biggest customers, and calculate the sales growth rate.

By analyzing the data, we aim to answer the following questions:

Best-Selling Products: Which types of paper are the most popular among the customers? By identifying the best-selling products, we can focus on optimizing their production, ensuring sufficient stock levels, and exploring potential opportunities for product diversification.

Biggest Customers: Which companies contribute the most to the overall sales? Identifying the biggest customers allows us to develop targeted strategies to nurture and strengthen these relationships. It also provides insights into the preferences and purchasing behaviour of key clients.

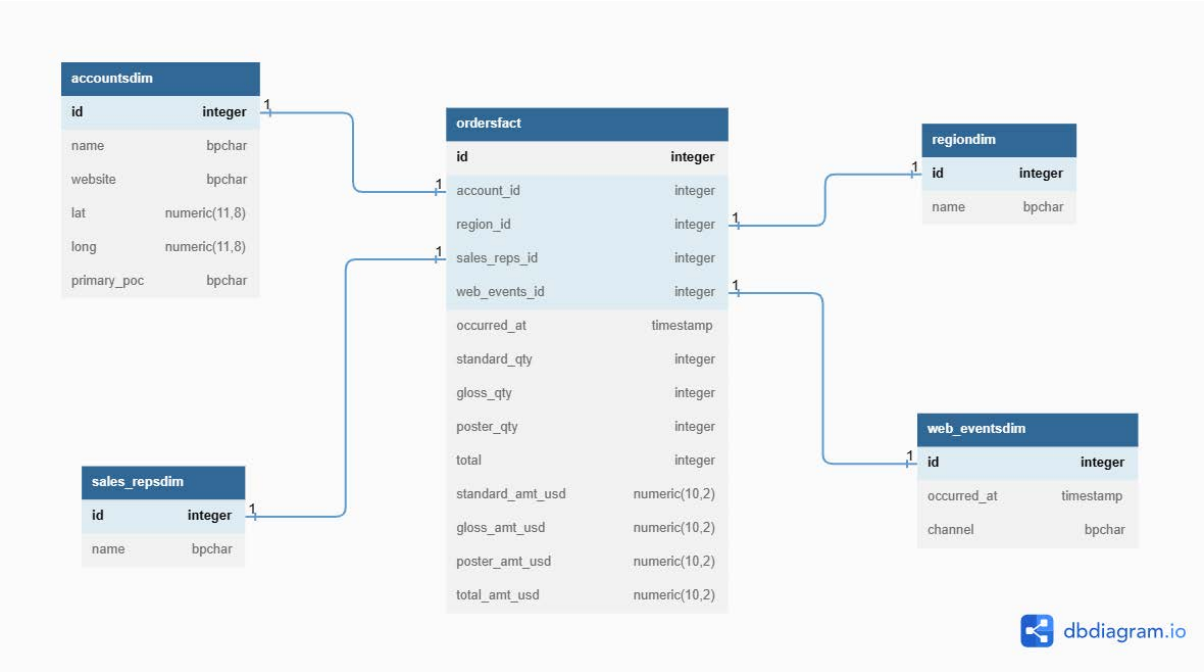
Sales Growth Rate: How has the sales performance evolved over time? Calculating the sales growth rate enables us to measure the company's progress and identify trends. By understanding the growth trajectory, we can make informed decisions regarding resource allocation, marketing efforts, and expansion plans.

- *Along with the above-mentioned objectives, we can analyse more from the dataset and explore few other outcomes as well. There are other insights for every queries as well.*

Through these analyses, we can gain a deeper understanding of the market dynamics, optimize our product offerings, and develop strategies to enhance customer satisfaction and sales growth. This information will help drive data-driven decision-making processes and support the overall success of the business.

In this database, we have records of orders for different types of paper placed by companies such as Walmart, Microsoft, among others. We can see how much of each type of paper was ordered, how much was spent, who was responsible for the order, in which region the company is located, and the dates of the different web events each company has conducted.

Relational Schema



Datasets used:

- **accounts**: This table contains all the different companies, their id (account_id), website, and contact of point.
- **orders**: Timestamp of every order, the quantity ordered of every type of paper (standard_qty, gloss_qty, poster_qty), the total, how much money was spent in each type of paper (standard_amt_usd, gloss_amt_usd, poster_amt_usd) and the total in dollars.
- **region**: Four regions: Northeast, Midwest, Southeast, West
- **sales_reps**: This table shows all the sales representative names with their corresponding id and name.
- **web_events**: All the web events conducted by each company, the account_id, the date each web event was conducted and the channel (Facebook, Twitter, etc)

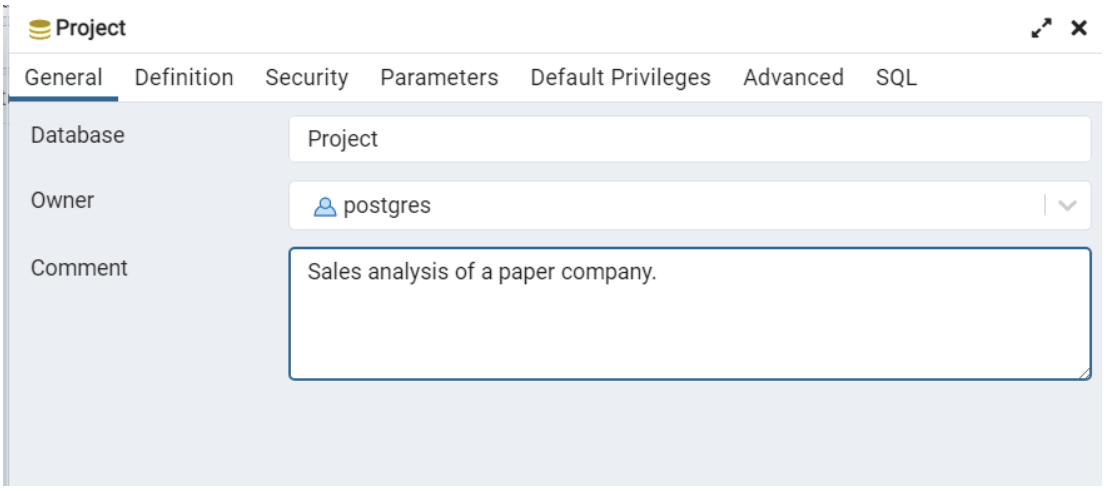
Creating a database 'Project' with five tables comprising one fact table and four dimension tables

Fact Table : Orders Table

Dimension Table : Region, Sales Representative, Web Events, Account Tables

Queries:

- Creating a database naming ‘Project’ for sales analysis.



- Creating tables under Project database

1. Region Table

Query Query History

```
/* Creating Region table to store the regions of companies */


CREATE TABLE region                -- Table is named as region
(
    id integer not null,           -- region id with integer datatype with not null constraint
    name bpchar,                   -- region name in blank padded characters
    primary key(id)                -- Defining region id as pimary key (PK)
);
```

2. Sales Representative Table

Query Query History

```
/* Creating Sales Representatives Table to know their details */

CREATE TABLE sales_reps                    -- Table is named as sales_reps
(
    id integer not null,                   -- The sales_reps id with integer datatype with not null constraint
    name bpchar,                           -- The sales_reps name with blank padded character datatype
    primary key(id)                        -- Defining the sales_rep id as the primary key (PK)
);
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 49 msec.
```

3. Web events Table.

```
/* Creating Web events table to store details */

CREATE TABLE web_events                    -- Table is named as web_events
(
    id integer not null,                   -- The web_events id with integer datatype with not null constraint
    occurred_at timestamp,                 -- Occurred_at column store data in timestamp datatype which stores data&time.
    channel bpchar,                        -- Web channel as channel with blank padded character datatype
    primary key(id)                        -- Defining web events id as primary key (PK)
);
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 44 msec.
```

4. Accounts Table

```
/* Creating Accounts table to store details */

CREATE TABLE accounts                      -- Table is named as accounts
(
    id integer ,                   -- The accounts id with integer datatype with not null constraint
    name bpchar,                           -- The accounts name with blank padded character datatype
    website bpchar,                        -- Website with blank padded character datatype
    lat numeric(11,8),                     -- Latitude with numeric datatype with precision and scale value
    long numeric(11,8),                    -- Longtitude with numeric datatype with precison and scale value
    primary_poc bpchar,                    -- Point of Contact with blank padded character datatype
    primary key(id)                        -- Defining accounts id as Primary key (PK)
);
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 129 msec.
```

5. Orders Table

Query Query History

```
/* Creating Orders table comprising details of other tables */
-- Orders table is created with multiple columns serving as a foreign key, which also stores the different types of papers' qunatity
--And the amount spent on the varieties of paper type.

CREATE TABLE orders
(
    id integer not null ,
    account_id integer not null,
    region_id integer not null,
    sales_reps_id integer not null,
    web_events_id integer not null,
    occurred_at timestamp,
    standard_qty integer,
    gloss_qty integer,
    poster_qty integer,
    total integer,
    standard_amt_usd numeric(10,2),
    gloss_amt_usd numeric(10,2),
    poster_amt_usd numeric(10,2),
    total_amt_usd numeric(10,2),
    primary key(id),
    constraint fk_orders_accounts foreign key(account_id)
    references accounts(id)
    constraint fk_orders_region foreign key(region_id)
    references region(id)
    constraint fk_orders_salesreps foreign key(sales_reps_id)
    references sales_reps(id)
    constraint fk_orders_webevents foreign key(web_events_id)
    references web_events(id)
```

Data Output Messages Notifications

Total rows: 0 of 0 Query complete 00:00:00.043

Populating the Tables:

- Inserting each and every table using insert command.
- Datasets was imported from different sources for convenience
- With necessary changes to datasets, all the five tables including fact table and dimension table is populated with data.

Queries:

1. First, let's see how much we sale per year?

   Selecting year using extract function & giving an alias name,

   and sum of the total amount used using aggregate function sum () and giving an alias name,

   from the orders table and grouping it by the year and sorting it by the total amount ascendingly

   Query and output for this question:

Query Query History

```
/* First, lets see how much we sale per year */

/* Selecting year using extract function & giving an alias name,
   and sum of the total amount used using aggregate function sum() and giving an alias name,
   from the orders table and grouping it by the year and ordering it by the total amount ascendingly */

SELECT EXTRACT(YEAR FROM occurred_at) AS year,
SUM(total_amt_usd) AS total_usd
FROM orders
GROUP BY year
ORDER BY total_usd ASC;
```

Data Output Messages Notifications

	year numeric	total_usd numeric
1	2014	6748.54
2	2015	63928.43
3	2016	125164.06

Insights:

- We observe that 2014 has the least sales among the three years, but sales have been increasing year after year with large growth percentage.

2. Providing the region for each sales_reps along with their associated accounts for the 'Midwest' region only.

- As we want the region name, the sales reps name and the account name, we have to work with 3 tables: regions, sales_reps and accounts
- So, we select the name column from the regions table, the name column from the sales_reps table and the name column from the accounts table
- Then we join the region table with the sales_reps table and that with the account tables
- As we only want the Midwest region, we filter with a where clause
- And finally, we sort by the account name

Query and output for this question:

```
/*  Providing the region for each sales_rep along with their associated accounts for the 'Midwest' region only.*/

--As we want the region name, the sales rep name and the account name, we have to work with 3 tables: regions, sales_reps and accounts
--So we select the name column from the regions table, the name column from the sales_reps table and the name column from the accounts table
--Then we join the region table with the sales_reps table and that with the account tables
-- As we only want the Midwest region, we filter with a where clause
-- And finally we sort by the account name

SELECT r.name as Region, sr.name as Rep_name, acc.name as account_name
FROM orders ord join region r on r.id=ord.region_id
                join sales_reps sr on sr.id=ord.sales_reps_id
                join accounts acc on acc.id=ord.account_id
WHERE r.name='Midwest'
ORDER BY acc.name ASC
```

	region character	rep_name character	account_name character
45	Midwest	Maren Musto	UnitedHealth Group
46	Midwest	Dorotha Seawell	UnitedHealth Group
47	Midwest	Moon Torian	UnitedHealth Group
48	Midwest	Earlie Schleusner	UnitedHealth Group
49	Midwest	Sibyl Lauria	Verizon
50	Midwest	Julia Behrman	Verizon
51	Midwest	Necole Victory	Walmart
52	Midwest	Sibyl Lauria	Walmart

Total rows: 63 of 63 Query complete 00:00:00.061

Insights:

- We were able to know the sales representative's names with the respective company by their account name corresponding the Midwest region. Here, using filter, we can further look down for other sales representatives in the other regions.

3. Let's see the percentage of growth in each year in the last quarter.

- We extract the year of the occurred_at column from the orders table, we extract the month, and we also extract the day.
- Apply the sum() function to the total_amt_usd also from the orders table and give an alias name
- We filter by where clause with month and day by any conditions.
- We group by year, month and day and sort it by the total amount used.

Query and output for this question:

```sql
-- 3.Lets see the percentage of growth in each year in the last quarter.

--We extract the year of the occurred_at column from the orders table ,we  extract the month, and we also extract the day.
--Apply the sum() function to the total_amt_usd also from the orders table and give an alias name
--We filter by where clause with month and day by any conditions.
--We group by year,month and day and sort it by the total amount used.

WITH CTE_GROWTH AS
(SELECT EXTRACT(YEAR FROM occurred_at) AS year,
        EXTRACT(MONTH FROM occurred_at) AS month,
        EXTRACT(DAY FROM occurred_at) AS day,
        SUM(total_amt_usd) AS total_usd
FROM orders
WHERE EXTRACT(MONTH FROM occurred_at) IN (9) AND EXTRACT(DAY FROM occurred_at) = 1
GROUP BY year, month, day
ORDER BY month asc )

SELECT year, month, day,total_usd,
total_usd - LAG(total_usd) OVER (ORDER BY year ASC) AS growth,
(total_usd - LAG (total_usd) OVER (ORDER BY year ASC))/LAG (total_usd) OVER (ORDER BY year ASC)*100 AS percentage_growth
FROM CTE_GROWTH
```

Data Output Messages Notifications

	year numeric	month numeric	day numeric	total_usd numeric	growth numeric	percentage_growth numeric
1	2015	9	1	1061.33	[null]	[null]
2	2016	9	1	2820.25	1758.92	165.7279074368952200

Insights:

- From 2015 to 2016 our sales grew by 165% when compared on that particular month of September and that specific first day. The cells in the tells that the percentage rose hugely on a specific day resulting in a huge growth and growth percentage.

4. For each account, determining the average amount of each type of paper they purchased across their orders.

- We need the column name of the accounts table, and the mean of each type of paper each one of the accounts purchased
- Thus, we apply the aggregate function avg () in each type of paper
- As we want the account name and the mean of each type of paper each one of the accounts purchased across their orders, we join the accounts table with the orders table
- Lastly, we group by the account_name
- We're creating a view to store the query and later retrieving the output from the view by using round() to some columns.

Query and output for this question:

```sql
/* For each account,the average amount of each type of paper they purchased across their orders.*/

--We need the column name of the accounts table, and the mean of each type of paper each one of the accounts purchased
--Thus, we apply the aggregate function avg() in each type of paper
--As we want the account name and the mean of each type of paper each one of the accounts purchased across their orders
--we join the accounts table with the orders table
--Lastly we group by the account_name

CREATE VIEW  average_amount AS (
    SELECT ac.name AS account_name, AVG(o.standard_qty) AS average_standard_qty, AVG(o.gloss_qty) AS average_gloss_qty,
    AVG(o.poster_qty) AS average_poster_qty, AVG(total) as average_total
    FROM accounts ac JOIN orders o ON ac.id=o.account_id
    GROUP BY ac.name
    ORDER BY average_standard_qty DESC )

SELECT account_name , ROUND(average_standard_qty,2) AS average_standard_qty ,
ROUND(average_gloss_qty,2) AS average_gloss_qty ,ROUND(average_poster_qty,2) AS average_poster_qty,
ROUND(average_total,2) AS average_total
FROM average_amount;
```

Data Output Messages Notifications

	account_name character	average_standard_qty numeric	average_gloss_qty numeric	average_poster_qty numeric	average_total numeric
1	Berkshire Hathaway	1148.00	0.00	215.00	1363.00
2	General Motors	803.00	31.00	19.00	853.00
3	Exxon Mobil	527.00	14.00	0.00	541.00
4	Apple	514.67	17.83	11.33	543.83
5	UnitedHealth Group	496.50	34.83	5.00	536.33
6	AmerisourceBergen	325.00	59.00	11.50	395.50
7	Ford Motor	314.09	32.50	34.73	381.32
8	Verizon	292.50	31.07	20.43	344.00
9	CVS Health	276.33	25.67	19.33	321.33
10	McKesson	254.00	28.00	30.00	312.00
11	AT&T	151.96	14.18	16.86	183.00
12	General Electric	146.75	39.00	12.25	198.00
13	Walmart	113.13	45.31	39.25	197.69

Insights:

- When the created view gets executed, we were able to determine the average amount of each type of paper each company has purchased across their order by the account name.
- The company that has bought us more quantity of paper is 'Berkshire Hathaway'.

5. For each account, determining the average amount spent per order on each paper type.

- Selecting the name of the accounts from accounts table, &the mean of the amount spent on each paper type by the respective accounts
- Thus, we apply the aggregate function avg () on the amount spent on each paper type.
- As we want the account name and the mean of the amount spend on each type of paper, we join the accounts table with the orders table
- Lastly, we group by the account_name and sort it.

Query and output for this question:

```
/* 5.For each account, determining the average amount spent per order on each paper type. */

--Selecting the name of the acccounts from accounts table, &the mean of the amount spent on each paper type by the respective acocounts
--Thus, we apply the aggregate function avg() on the amount spent on each paper type.
--As we want the account name and the mean of the amount spend on each type of paper
--we join the accounts table with the orders table
--Lastly we group by the account_name and sort it.

SELECT ac.name AS account_name,
AVG(o.standard_amt_usd) AS avg_standard_amt_usd,
AVG(o.gloss_amt_usd) AS avg_gloss_amt_usd,
AVG(o.poster_amt_usd) AS avg_poster_amt_usd,
AVG(o.standard_amt_usd)+AVG(o.gloss_amt_usd)+AVG(o.poster_amt_usd) as total
FROM accounts ac JOIN orders o ON ac.id=o.account_id
GROUP BY ac.name
ORDER BY total DESC
```

Data Output Messages Notifications

	account_name character	avg_standard_amt_usd numeric	avg_gloss_amt_usd numeric	avg_poster_amt_usd numeric	total numeric
1	Berkshire Hathaway	5728.5200000000000000	0.00000000000000000000	1745.8000000000000000	7474.32000000000000000000
2	General Motors	4006.9700000000000000	232.1900000000000000	154.2800000000000000	4393.4400000000000000
3	Apple	2568.1866666666666667	133.5716666666666667	92.0266666666666667	2793.7850000000000001
4	UnitedHealth Group	2477.5350000000000000	260.9016666666666667	40.6000000000000000	2779.0366666666666667
5	Exxon Mobil	2629.7300000000000000	104.8600000000000000	0.00000000000000000000	2734.59000000000000000000
6	AmerisourceBergen	1621.7500000000000000	441.9100000000000000	93.3800000000000000	2157.0400000000000000
7	Ford Motor	1567.3136363636363636	243.4250000000000000	281.9863636363636364	2092.7250000000000000
8	Verizon	1459.5750000000000000	232.7250000000000000	165.8800000000000000	1858.1800000000000000
9	CVS Health	1378.9033333333333333	192.2433333333333333	156.9866666666666667	1728.1333333333333333
10	McKesson	1267.4600000000000000	209.7200000000000000	243.6000000000000000	1720.7800000000000000
11	Walmart	564.4937500000000000	339.3906250000000000	318.7100000000000000	1222.5943750000000000
12	General Electric	732.2825000000000000	292.1100000000000000	99.4700000000000000	1123.8625000000000000
13	AT&T	758.3017857142857143	106.1975000000000000	136.8800000000000000	1001.3792857142857143

Insights:

- From the table above we can know the list of companies and their spending on different types of paper and perform analysis.

- Also, if we are interested in a specific type of paper, we can sort the previous table; for example, let's say we are interested in standard paper, then we sort by the avg_standard_amt_usd column, and in this way, we can see who has spent the most on that type of paper.

6.Using the above question, find the account that had spent the maximum.

- Using the above in a Subquery we find the account of the company that has spent the most.
- The Subquery here is given an alias name 'A' and limiting it by 1 to know the maximum spent account.
- In this query we can visualize the account_name that had spent the most.

Query and output for this question:

```
/* 6. Using the above question , find the account that had spend the maximum */

-- Using the above in a Subquery we find the the account of the company that has spent the most.
--The Subquery here is given an alias name 'A' and limitting it by 1 to know the maximum spent account.
--In this query we can visualize the account_name that had spend the most

SELECT account_name, ROUND(avg_standard_amt_usd) AS avg_standard_amt_usd,ROUND(avg_gloss_amt_usd) AS avg_gloss_amt_usd,
ROUND(avg_poster_amt_usd) AS avg_poster_amt_usd,ROUND(total) AS total
FROM (  SELECT ac.name AS account_name, AVG(o.standard_amt_usd) AS avg_standard_amt_usd, AVG(o.gloss_amt_usd) AS avg_gloss_amt_usd,
        AVG(o.poster_amt_usd) AS avg_poster_amt_usd, AVG(o.standard_amt_usd)+AVG(o.gloss_amt_usd)+AVG(o.poster_amt_usd) as total
        FROM accounts ac JOIN orders o ON ac.id=o.account_id
        GROUP BY ac.name
        ORDER BY total DESC) A
LIMIT 1;
```

Data Output Messages Notifications

	account_name character	avg_standard_amt_usd numeric	avg_gloss_amt_usd numeric	avg_poster_amt_usd numeric	total numeric
1	Berkshire Hathaway	5729	0	1746	7474

Insights:

- From the table, we observed that *Berkshire Hathaway* is the company who has bought us the most specific to standard type of paper and also the total amount spent.
- Though Berkshire Hathaway constitute the most spending in purchasing the papers, it still remains lowest in gloss type of paper. When performed using filter (for gloss type) we will attain another company who tops the list with respect to its type.

7. Determining the number of times a particular channel was used in the web_events table for each sales rep. Ordering the table with the highest number of occurrences first.

- We want the sales reps name, the channel and the number of occurrences of these channels
- To attain those, we will work with web_events table, the sales_reps table and the orders table.
- Therefore, selecting the name column from the sales_reps table, the channel column from the web_events table, and apply the count () function to the channels to get the number of occurrences.
- Then, as we are working with the web_events and sales_reps tables we want to join them.
- We'll join them using orders table which is the only common between the tables.
- We then group by the sales reps name and the channel and finally sort by the number of occurrences

Query and output for this question:

```
/* 7. Determining the number of times a particular channel was used in the web_events table for each sales rep.
Order your table with the highest number of occurrences first.*/

--We want the sales rep name, the channel and the number of occurrences of these channels
--To attain those we will work with web_events table, the sales_reps table and the orders table.
--Therefore,selecting the name column from the sales_reps table, the channel column from the web_events table,
--and apply the count() function to the channels to get the number of occurences.
--Then, as we are working with the web_events and sales_reps tables we want to join them.
--We'll join them using orders table which is the only common between the tables.
--We then group by the sales rep name and the channel and finally  sort by the number of ocurrences

SELECT sr.name AS sales_rep_name, we.channel AS channel, count(channel) AS number_of_occurrences
from web_events we join orders ord on we.id=ord.web_events_id
                   join sales_reps sr on sr.id=ord.sales_reps_id
GROUP BY sr.name, we.channel
ORDER BY number_of_occurrences DESC
```

Data Output Messages Notifications

	sales_rep_name character	channel character	number_of_occurrences bigint
1	Eugena Esser	adwords	3
2	Tia Amato	adwords	2
3	Marquetta Laycock	direct	2
4	Ernestine Pickron	direct	2
5	Sherlene Wetherington	adwords	2
6	Samuel Racine	direct	2

Total rows: 102 of 102 Query complete 00:00:00.081

Insights:

- From this result, we can observe the effectiveness of the web channels for the sales representatives.
- With knowing which channel is effective for an individual, we can either focus on the effective channel, or find ways to improve in the other existing channels as well.

8. Provide the name for each region for every order, as well as the account name and the unit price they paid (total_amt_usd/total) for the order. However, only provide the results if the standard order quantity exceeds 100 and the poster order quantity exceeds 50.

- Here we sort the table to visualize the account name with the corresponding unit price
- In order to avoid a division by zero error, adding .01 to the denominator

Query and output for this question:

```
/* 8. Provide the name for each region for every order, as well as the account name and the unit price they paid (total_amt_usd/total)
for the order. However, only provide the results if the standard order quantity exceeds 100 and the poster order quantity
exceeds 50. */

--Here we sort the table to visualize the account name with the highest unit price
--In order to avoid a division by zero error, adding .01 to the denominator

SELECT r.name as region, ac.name as account_name, round((o.total_amt_usd/(o.total + 0.01)),2) as unit_price
FROM region r join orders o on r.id=o.region_id
              join accounts ac on ac.id=o.account_id
WHERE o.standard_qty > 100 AND o.poster_qty > 50
ORDER BY unit_price ASC
```

Query Query History

Data Output Messages Notifications

	region character	account_name character	unit_price numeric
1	Midwest	Berkshire Hathaway	5.48
2	Northeast	Ford Motor	5.51
3	Midwest	McKesson	5.71
4	Northeast	Ford Motor	5.72
5	Midwest	Ford Motor	5.91
6	Midwest	AT&T	5.96
7	Northeast	Walmart	5.97
8	Northeast	Verizon	6.04
9	Northeast	AT&T	6.27
10	Midwest	Ford Motor	6.28
11	Midwest	Walmart	6.63

Insights:

- From this observation, we can understand and learn about the unit prices of companies across regions.
- This result shows that the lowest unit price 5.48 USD.
- We can further perform analysis to know the unit prices in a particular region and can compare the companies from different regions.

9. Using the above question, display only the account name with highest unit price

- Here we sort the table to visualize the account name with the highest unit price using DESC

Query and output for this question:

```
/* 9. Using the above question, display only the account name with HIGHEST unit price */

--Here we sort the table to visualize the account name with the highest unit price using DESC

SELECT r.name as region, ac.name as account_name, round((o.total_amt_usd/(o.total + 0.01)),2) as unit_price
FROM region r join orders o on r.id=o.region_id
              join accounts ac on ac.id=o.account_id
WHERE o.standard_qty > 100 AND o.poster_qty > 50
ORDER BY unit_price DESC
LIMIT 1;
```

Data Output Messages Notifications

	region character	account_name character	unit_price numeric
1	Midwest	Walmart	6.63

Insights:

- With reference to previous queries, we did get the highest unit price across regions in this observation.
- By performing limiting, we attainted the highest unit price and its corresponding company and region.
- The highest unit price is 6.63 USD which corresponds to “Walmart” from the Midwest region.

10. To find out which paper has sold the most across companies.

- To find the quantity ordered of every type of paper.
- To do this, we have to perform sum () function on the paper types quantity to find the total.

Query and output for this question:

```
/* 10. To find out which paper has sold the most across companies and the total amount used on the respective types */

--To fine the quantity ordered of every type of paper
--To do this, we have to perform sum() function on the paper types quantity to find the total.

SELECT SUM(standard_qty) AS total_standard_qty,
SUM(gloss_qty) AS total_gloss_qty,
SUM(poster_qty) AS total_poster_qty
FROM orders
```

Data Output Messages Notifications

	total_standard_qty bigint	total_gloss_qty bigint	total_poster_qty bigint
1	29617	3380	2800

11. To find how much money was spent in each type of paper.

- To know the total amount used on the respective types.
- To do this, we have to perform sum () function on the amount spent in each paper type.

Query and output for this question:

```
/* 11.  To find how much money was spend in each type of paper */

--To do this,we have to perform sum() function on the amount spent in each paper type.

SELECT SUM(standard_amt_usd) AS total_standard_usd,
SUM(gloss_amt_usd) total_gloss_usd,
SUM(poster_amt_usd) AS  total_poster_usd
FROM orders
```

Data Output Messages Notifications

	total_standard_usd numeric	total_gloss_usd numeric	total_poster_usd numeric
1	147788.83	25316.20	22736.02

Insights:

- From the above two tables, we observe which paper was sold the most across companies and how much amount was used in the respective types.
- The most quantity sold across companies is standard type of paper.
- The most amount used a particular type of paper is, 147788.83 USD for standard type of paper.

12. In which month of which year did Walmart spend the most on gloss paper in terms of dollars?

- Selecting the year, month using extract () function, the total gloss amount using sum () from orders table and account name from accounts table
- We attain the required by joining these two tables on account id.
- To get the result specific to 'Walmart', we filter using WHERE clause
- Finally, we group by the year, month, name and sort it by the total gloss usd in descending order
- We limit the output to 1 to get which month had the highest spendings.

Query and output for this question:

```
/* 12. In which month of which year did Walmart spend the most on gloss paper in terms of dollars?*/

--Selecting the year,month using extract() function, the total gloss amount using sum() from orders table and account name from accounts
--We attain the required by joining these two tables on acccount id.
--TO get the result specific to 'Walmart', we filter using WHERE clause
--Finally we group by the year,name,name and sort it by the total gloss usd in descending order
--We limit the output to 1 to get which month had the highest spendings.

SELECT EXTRACT(YEAR FROM o.occurred_at) as year, EXTRACT( MONTH FROM o.occurred_at) AS MONTH,
ac.name AS account_name, SUM(gloss_amt_usd) AS total_gloss_usd
FROM orders o JOIN accounts ac
ON o.account_id=ac.id
WHERE ac.name='Walmart'
GROUP BY YEAR,MONTH,NAME
ORDER BY total_gloss_usd DESC
LIMIT 1
```

Data Output Messages Notifications

	year numeric	month numeric	account_name character	total_gloss_usd numeric
1	2016	10	Walmart	1071.07

Insights:

- To make analysis interesting, we performed queries to know which month of which year a company had spent the most amount in purchasing the gloss paper.
- From the above table, we attained the result for ‘Walmart’ where in the month of October in 2016, the company had spent the maximum amount of 1071.07 USD
- This analysis helps the company to understand their spendings and can be critical according to their financial breakdowns.
- These can be performed for every company by filtering which makes them aware.

13.Providing the region for each sales_reps along with their associated accounts. Provide only for accounts where the sales rep has a first name starting with S and in the Midwest region.

- The condition for the first name of the sales reps name should start with an S.
- Therefore, in the where clause we a condition

Query and output for this question:

```
/* 13.Providing the region for each sales_rep along with their associated accounts.
This time only for accounts where the sales rep has a first name starting with S and in the Midwest region.*/

--The condition for the first name of the sales rep name should starts with an S.
--Therefore, in the where clause we a condition

SELECT r.name as Region, sr.name as Rep_name, ac.name as account_name
FROM orders o JOIN region r ON r.id=o.region_id
              JOIN sales_reps sr ON sr.id=o.sales_reps_id
              JOIN accounts ac ON o.account_id=ac.id
WHERE r.name='Midwest' AND sr.name LIKE 'S%'
ORDER BY ac.name ASC
```

Data Output Messages Notifications

	region character	rep_name character	account_name character
1	Midwest	Sherlene Wetherington	Apple
2	Midwest	Saran Ram	AT&T
3	Midwest	Saran Ram	CVS Health
4	Midwest	Shawanda Selke	Ford Motor
5	Midwest	Sibyl Lauria	Ford Motor
6	Midwest	Samuel Racine	Ford Motor
7	Midwest	Sibyl Lauria	Verizon
8	Midwest	Silvana Virden	Walmart
9	Midwest	Sibyl Lauria	Walmart
10	Midwest	Shawanda Selke	Walmart

Insights:

- From the above table, we attained names starting with 'S' in Midwest region.
- This analysis can be played in any required manner. For Midwest region, you can get the names of representatives of your choice or your requirement.

14. To find the maximum amount spent by a particular account across the years and its corresponding name & id.

- To do this, we have to join orders and accounts tables.
- To find the specific company that had spent the maximum, we have to use sum () function for total_amt_usd.
- The sum () is used in a subquery to find the maximum and finally, the result is grouped by name & id.

Query and output for this question:

```
/* 14. To find the maximum amount spent by a particular account across the years and it's corresponding name & id  */

--To do this,we have to join orders and accounts tables.
--To find the specific company that had spent the maximum, we have to use sum() function for total_amt_us
--The sum() is used in an subquery to find the maximum and finally, the result is grouped by name & id.

SELECT o.account_id,ac.name AS account_name,SUM(total_amt_usd) AS Total_Amount
FROM orders o JOIN accounts ac ON o.account_id=ac.id
GROUP BY account_id,ac.name
HAVING SUM(total_amt_usd) >=
ALL(SELECT SUM(total_amt_usd) FROM orders GROUP BY account_id);
```

Data Output Messages Notifications

	account_id integer	account_name character	total_amount numeric
1	1081	Ford Motor	46039.93

Insights:

- From this table, we can observe that 'Ford Motor' had spent the maximum amount across all the years.

15. With respective to 2016, determining the total quantity of papers

and the total amount used on all types across region.

- Firstly, we have to join region and orders table to relate the paper details with region.
- By extracting year from occurred_at, using SUM () function we can find the total quantity and total amount.
- Finally, we can filter year using WHERE clause and group it by regions.

```
/* 15. With respective to 2016 , determining the total quantity of papers
       and the total amount used on all types across region.*/

-- Firstly, we have to join region and orders table to relate the paper details with region.
--By extracting year from occurred_at, using SUM () function we can find the total quantity and total amount.
--Finally, we can filter year using WHERE clause and group it by region.


SELECT r.name AS region_name ,EXTRACT(YEAR FROM occurred_at) AS year,
ROUND(SUM(total)) AS Total_quantity,
ROUND(SUM(total_amt_usd)) AS Total_amount
FROM orders o JOIN region r
ON o.region_id=r.id
WHERE EXTRACT(YEAR FROM occurred_at) = 2016
GROUP BY r.name,year
```

Data Output Messages Notifications

	region_name character	year numeric	total_quantity double precision	total_amount numeric
1	Midwest	2016	20532	112160
2	Northeast	2016	786	4478
3	Southeast	2016	696	3743
4	West	2016	903	4783

Insights:

- From this observation, we can understand that the reason for the highest sales in 2016 is because of the Midwest region.
- Midwest region solely contributes around 90% of the sales in that particular year with other regions together comprises around 10% .

Conclusions:

Our primary goals have been achieved, which summarises:

1**. Sales growth rate:** Year-on-year there has been a steady increase in sales with a surge of over 100% from 2015 to 2017.

2. **Highest Sales**: We expect 2016 to have the largest sales.

3. **Top company:** Our biggest client is *Berkshire Hathaway* followed by *General Motors*

4. **Best-selling products**: The paper that has sold the most is standard type paper.

5. The highest unit price is 6.63 USD, and lowest unit price is 5.48 USD

- Along with the primary goals achieved, from each query there were lot of insights to learn and analyse. Each of those had different dimensions of analysis, which helps us better to understand data and make decisions.
- Thus, all our goals are met and the analysis performed will be helpful for taking data driven decision.

APPENDIX:

Performing the queries for various analyis and Creating Table queries (end)

1. First, let's see how much we sale per year

```
SELECT EXTRACT(YEAR FROM occurred_at) AS year,
SUM(total_amt_usd) AS total_usd
FROM orders
GROUP BY year
ORDER BY total_usd ASC
```

2. Providing the region for each sales_rep along with their associated accounts for the 'Midwest' region only.

```
SELECT r.name AS Region, sr.name AS Rep_name, acc.name AS account_name
FROM orders ord JOIN region r ON r.id=ord.region_id
JOIN sales_reps sr ON sr.id=ord.sales_reps_id
JOIN accounts acc ON acc.id=ord.account_id
WHERE r.name='Midwest'
ORDER BY acc.name ASC
```

3. Let's see the percentage of growth in each year in the last quarter.

```
WITH CTE_GROWTH AS
(SELECT EXTRACT(YEAR FROM occurred_at) AS year,
                EXTRACT(MONTH FROM occurred_at) AS month,
                EXTRACT(DAY FROM occurred_at) AS day,
                SUM(total_amt_usd) AS total_usd
FROM orders
WHERE EXTRACT(MONTH FROM occurred_at) IN (9) AND EXTRACT(DAY FROM occurred_at) = 1
GROUP BY year, month, day
ORDER BY month ASC ) ;


SELECT year, month, day, total_usd,
total_usd - LAG(total_usd) OVER (ORDER BY year ASC) AS growth,
(total_usd - LAG (total_usd) OVER (ORDER BY year ASC))/LAG (total_usd) OVER (ORDER BY year
ASC)*100 AS percentage_growth
FROM CTE_GROWTH
```

4. For each account, determining the average amount of each type of paper they purchased across their orders.

```
CREATE VIEW  average_amount AS (

        SELECT ac.name AS account_name, AVG(o.standard_qty) AS average_standard_qty,
        AVG(o.gloss_qty) AS average_gloss_qty,

        AVG(o.poster_qty) AS average_poster_qty, AVG(total) as average_total

        FROM accounts ac JOIN orders o ON ac.id=o.account_id

        GROUP BY ac.name

        ORDER BY average_standard_qty DESC ) ;


-- View created and named as average_amount


SELECT account_name , ROUND(average_standard_qty,2) AS average_standard_qty ,

ROUND(average_gloss_qty,2) AS average_gloss_qty ,ROUND(average_poster_qty,2) AS
average_poster_qty,

ROUND(average_total,2) AS average_total

FROM average_amount
```

5. For each account, determining the average amount spent per order on each paper type.

```
SELECT ac.name AS account_name,

AVG(o.standard_amt_usd) AS avg_standard_amt_usd,

AVG(o.gloss_amt_usd) AS avg_gloss_amt_usd,

AVG(o.poster_amt_usd) AS avg_poster_amt_usd,

AVG(o.standard_amt_usd)+AVG(o.gloss_amt_usd)+AVG(o.poster_amt_usd) as total

FROM accounts ac JOIN orders o ON ac.id=o.account_id

GROUP BY ac.name

ORDER BY total DESC
```

6. Using the above question, find the account that had spent the maximum.

```
SELECT account_name, ROUND(avg_standard_amt_usd) AS avg_standard_amt_usd,
ROUND(avg_gloss_amt_usd) AS avg_gloss_amt_usd,

ROUND(avg_poster_amt_usd) AS avg_poster_amt_usd,ROUND(total) AS total

FROM ( SELECT ac.name AS account_name, AVG(o.standard_amt_usd) AS avg_standard_amt_usd,
        AVG(o.gloss_amt_usd) AS avg_gloss_amt_usd,

        AVG(o.poster_amt_usd) AS avg_poster_amt_usd,
        AVG(o.standard_amt_usd)+AVG(o.gloss_amt_usd)+AVG(o.poster_amt_usd) as total

        FROM accounts ac JOIN orders o ON ac.id=o.account_id

        GROUP BY ac.name

        ORDER BY total DESC) A

LIMIT 1;
```

7. Determining the number of times a particular channel was used in the web_events table for each sales rep. Order your table with the highest number of occurrences first.

```
SELECT sr.name AS sales_rep_name, we.channel AS channel, COUNT(channel) AS
number_of_occurrences

FROM web_events we

JOIN orders ord on we.id=ord.web_events_id

JOIN sales_reps sr on sr.id=ord.sales_reps_id

GROUP BY sr.name, we.channel

ORDER BY number_of_occurrences DESC
```

8. Provide the name for each region for every order, as well as the account name and the unit price they paid (total_amt_usd/total) for the order. However, only provide the results if the standard order quantity exceeds 100 and the poster order quantity exceeds 50.

```
SELECT r.name AS region, ac.name AS account_name, ROUND((o.total_amt_usd/(o.total + 0.01)),2)
AS unit_price

FROM region r JOIN orders o ON r.id=o.region_id

JOIN accounts ac ON ac.id=o.account_id

WHERE o.standard_qty > 100 AND o.poster_qty > 50

ORDER BY unit_price ASC
```

9. Using the above question, display only the account name with HIGHEST unit price.

```
SELECT r.name AS region, ac.name AS account_name, round((o.total_amt_usd/(o.total + 0.01)),2) AS
unit_price

FROM region r JOIN orders o ON r.id=o.region_id

JOIN accounts ac ON ac.id=o.account_id

WHERE o.standard_qty > 100 AND o.poster_qty > 50

ORDER BY unit_price DESC

LIMIT 1;
```

10. To find out which paper has sold the most across companies and the total amount used on the respective types.

```
SELECT SUM(standard_qty) AS total_standard_qty,
SUM(gloss_qty) AS total_gloss_qty,
SUM(poster_qty) AS total_poster_qty
FROM orders
```

11. To find how much money was spend in each type of paper.

```
SELECT SUM(standard_amt_usd) AS total_standard_usd,
SUM(gloss_amt_usd) total_gloss_usd,
SUM(poster_amt_usd) AS  total_poster_usd
FROM orders
```

12. In which month of which year did Walmart spend the most on gloss paper in terms of dollars?

```
SELECT EXTRACT(YEAR FROM o.occurred_at) as year, EXTRACT( MONTH FROM o.occurred_at) AS
MONTH, ac.name AS account_name, SUM(gloss_amt_usd) AS total_gloss_usd
FROM orders o JOIN accounts ac
ON o.account_id=ac.id
WHERE ac.name='Walmart'
GROUP BY YEAR,MONTH,NAME
ORDER BY total_gloss_usd DESC
LIMIT 1
```

13. Providing the region for each sales_rep along with their associated accounts. This time only for accounts where the sales rep has a first name starting with S and in the Midwest region.

```
SELECT r.name as Region, sr.name as Rep_name, ac.name as account_name
FROM orders o JOIN region r ON r.id=o.region_id
JOIN sales_reps sr ON sr.id=o.sales_reps_id
JOIN accounts ac ON o.account_id=ac.id
WHERE r.name='Midwest' AND sr.name LIKE 'S%'
ORDER BY ac.name ASC
```

14. To find the maximum amount spent by a particular account across the years and it's corresponding name & id.

```
SELECT o.account_id,ac.name AS account_name,SUM(total_amt_usd) AS Total_Amount
FROM orders o JOIN accounts ac ON o.account_id=ac.id
GROUP BY account_id,ac.name
HAVING SUM(total_amt_usd) >=
ALL(SELECT SUM(total_amt_usd) FROM orders GROUP BY account_id);
```

15. With respective to 2016, determining the total quantity of papers and the total amount used on all types across region.

```
SELECT r.name AS region_name ,EXTRACT(YEAR FROM occurred_at) AS year,
ROUND(SUM(total)) AS Total_quantity,
ROUND(SUM(total_amt_usd)) AS Total_amount
FROM orders o JOIN region r
ON o.region_id=r.id
WHERE EXTRACT(YEAR FROM occurred_at) = 2016
GROUP BY r.name,year
```

CREATING TABLES:

- **Creating Sales Representatives Table**

```
CREATE TABLE sales_reps
(
        id integer not null,
        name bpchar,
        primary key(id)
);
```

- **Creating Accounts table**

```
CREATE TABLE accounts
(
        id integer ,
        name bpchar,
        website bpchar,
        lat numeric(11,8),
        long numeric(11,8),
        primary_poc bpchar,
        primary key(id)
);
```

- **Creating Web events table**

```
CREATE TABLE web_events
(
        id integer not null,
        occurred_at timestamp,
        channel bpchar,
        primary key(id)
);
```

- **Creating Region Table**

```
CREATE TABLE region (
        id integer,
        name bpchar,
        primary key(id)
);
```

- **Creating Orders table**

```
CREATE TABLE orders
(
        id integer not null ,
        account_id integer not null,
        region_id integer not null,
        sales_reps_id integer not null,
        web_events_id integer not null,
        occurred_at timestamp,
        standard_qty integer,
        gloss_qty integer,
        poster_qty integer,
        total integer,
        standard_amt_usd numeric(10,2),
        gloss_amt_usd numeric(10,2),
        poster_amt_usd numeric(10,2),
        total_amt_usd numeric(10,2),
        primary key(id),
        constraint fk_orders_accounts foreign key(account_id)
        references accounts(id),
        constraint fk_orders_region foreign key(region_id)
        references region(id),
        constraint fk_orders_salesreps foreign key(sales_reps_id)
        references sales_reps(id),
        constraint fk_orders_webevents foreign key(web_events_id)
        references web_events(id)
);
```