Views Overview: Introduction to views as virtual tables derived from the result of a SELECT query.

What is a View in SQL?

A view is a virtual table whose contents are obtained from an existing table or tables in a database. The contents of a view are derived from the result of a SELECT query.

View defines a customized query that retrieves data from one or more tables, and represents the data as if it was coming from a single source.

A view is not a real table with data instead, it is a virtual table created from a **SELECT** query (can be thought of as a reflection) and this shows a subset of data from one of more table.

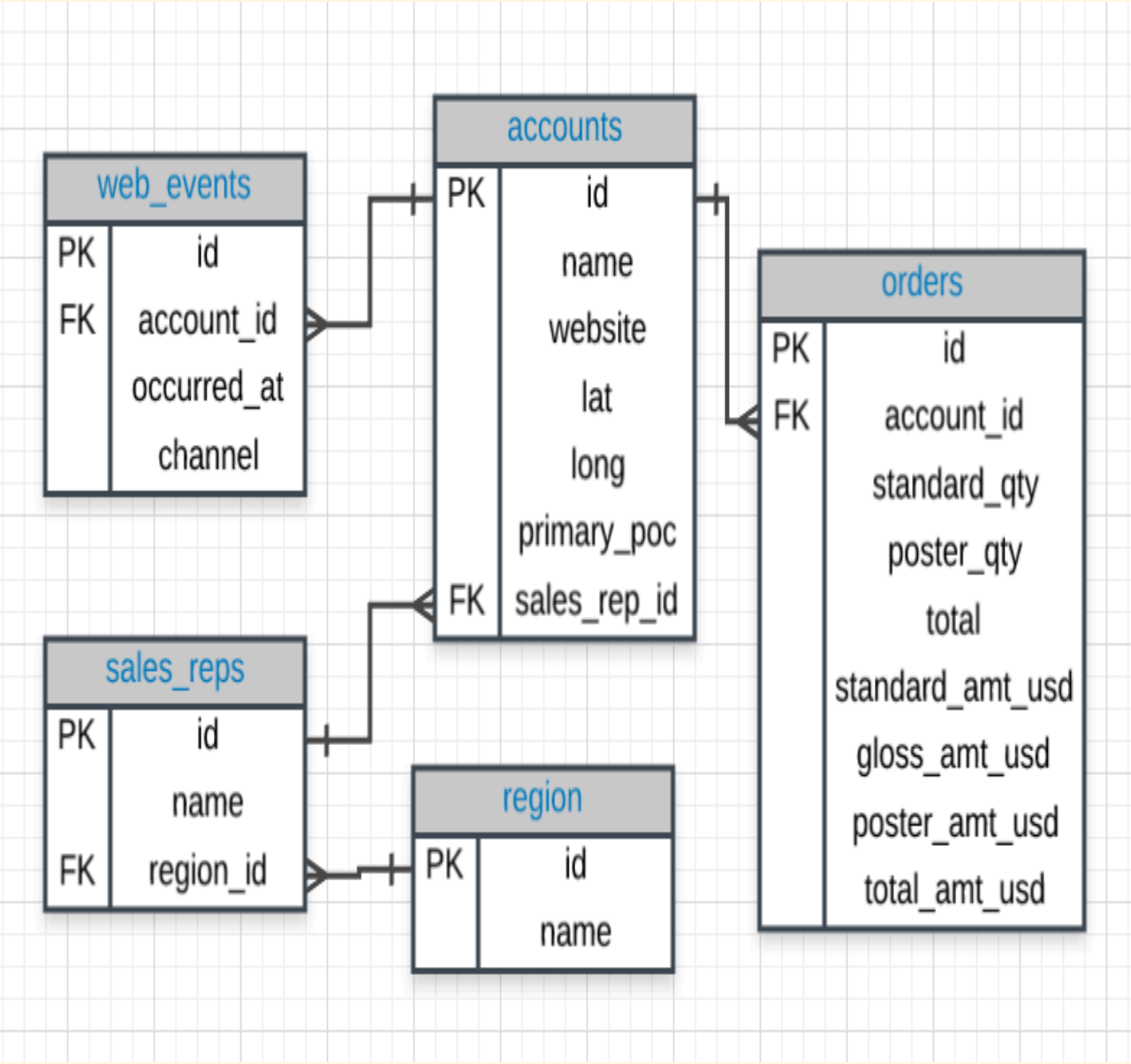
ABOUT THE DATABASE

I will be using the ***Parch and Posey*** database for the examples in this presentation.

The Parch and Posey database is a widely-used fictional database, serving as a teaching tool for database management. It offers a comprehensive structure with tables representing different business aspects like customers, orders, and products. It's a valuable resource for honing database management skills and understanding real-world scenarios.

ENTITY RELATIONSHIP DIAGRAM (ERD) FOR PARCH AND POSEY DATABASE

The image to the side defines the relationship between the tables of the database



How to Create a View

Below is a simple syntax for creating *Views* in SQL from a single table.

```
CREATE VIEW view_name AS
SELECT column1, column2...
FROM table
WHERE condition
ORDER BY condition;
```

- **CREATE VIEW**: this is the statement used in creating the view.
- **view_name**: the name for the view.

Example of a Simple View

```sql
--- This code is to get all data from the Orders table where the year is 2016
CREATE VIEW Orders2016 AS
SELECT *
FROM orders
WHERE EXTRACT(YEAR FROM occurred_at) = 2016;
```

From the image above, I created a view named *Orders2016* which contains all fields from the orders table where the year the order was made is 2016.

Let's see what the view looks like.

```sql
/*Select all feilds from the orders2016 view and
retrieving the first 5 rows*/
SELECT *
FROM orders2016
LIMIT 5;
```

The query above is to get all the flieds from the *Orders2016* view and retrievinf the first five rows on the data.

5 rows returned

	id integer	account_id integer	occurred_at timestamp without time zone	standard_qty integer	gloss_qty integer	poster_qty integer	total integer	standard_amt_usd numeric	gloss_amt_usd numeric	poster_amt_usd numeric	total_amt_usd numeric
1	4	1001	2016-01-02 01:18:24	144	32	0	176	718.56	239.68	0.00	958.24
2	5	1001	2016-02-01 19:27:27	108	29	28	165	538.92	217.21	227.36	983.49
3	6	1001	2016-03-02 15:29:32	103	24	46	173	513.97	179.76	373.52	1067.25
4	7	1001	2016-04-01 11:20:18	101	33	92	226	503.99	247.17	747.04	1498.20
5	8	1001	2016-05-01 15:55:51	95	47	151	293	474.05	352.03	1226.12	2052.20

Syntax for creating a *View* from two (2) table

```
CREATE VIEW view_name AS
SELECT column1, column2...
FROM table1
JOIN table2 ON table1.id = table2.id
WHERE condition
ORDER BY condition;
```

- **JOIN**: keyword for combining tables.
- **ON**: clause used with the JOIN keyword to combine tables.

A view can be created from one or more tables.

```sql
--- get the names of all employees in the Midwest region
CREATE VIEW reps_region AS
SELECT sales_reps.name AS SalesRepName, region.name AS RegionName
FROM sales_reps
JOIN region
ON sales_reps.region_id = region.id
WHERE region.name = 'Midwest';
```

For this example, I created a view named *reps_region*, retrieved the name of the Sales representative located in the *Midwest* region (the WHERE clause).

To see what the data in the *reps_region* view looks like.

```sql
--- data in reps_region view
SELECT *
FROM reps_region;
```

After running the query above it returns the output by the side.

It shows that there are only nine sales representatives located in the **Midwest** region.

9 rows returned

	name bpchar	name bpchar
1	Sherlene Wetherington	Midwest
2	Chau Rowles	Midwest
3	Carletta Kosinski	Midwest
4	Charles Bidwell	Midwest
5	Cliff Meints	Midwest
6	Delilah Krum	Midwest
7	Kathleen Lalonde	Midwest
8	Julie Starr	Midwest
9	Cordell Rieder	Midwest

WORKING WITH A VIEW

After creating a view, one can carryout analysis work on the created view. To see all data in the created view

```
SELECT * FROM view_name;
```

Upon retrieving data from the view, we can commence the analysis on the created view. This analysis encompasses various operations, including but not limited to calculating the **minimum**, **maximum**, **average**, **sum**, **count**, and **standard deviation**...

Using the previously created *Orders2016*, let's check the second half total of *standard*, *gloss* and *poster* sold and *total_amt_usd* made.

```sql
/* second half of the year report including the total amount of
units sold for each product and the amount made*/
SELECT SUM(standard_qty) AS TotalStandardQuantity,
    SUM(gloss_qty) AS TotalGlossQuantity,
    SUM(poster_qty) AS TotalPosterQuantity,
    SUM(total_amt_usd) AS TotalProductSum
FROM orders2016
WHERE EXTRACT(MONTH FROM occurred_at) > 5;
```

The image above demonstrates working with view, notice, after the FROM keyword comes the name of the view that was created earlier.

Below is the result of running the query from the earlier page. The query result the total unit of standard quantity, gloss quantity and poster quantity sold for the second half of the year. The last field of the output shows the total amount of money made in USD for the units sold in the second half of the year.

1 row returned

	totalstandardquantity bigint	totalglossquantity bigint	totalposterquantity bigint	totalproductsum numeric
1	750990	392605	311649	9218641.43

HOW TO DELETE A VIEW

While there is a syntax for creating views, there is also a syntax for deleting views. Below is the syntax for deleting view.

```
DROP VIEW view_name;
```

I decided to drop both views I created for explanation purposes.

```
1   --- drop a view
2   DROP VIEW orders2016;
3
4   DROP VIEW reps_region;
```

USES OF VIEW

- **Restricting data access**:
  - Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.
  - Useful in technical interviews for database administrator.
- **Reusability**:
  - Instead of repeatedly writing the same SELECT statements, we create a view once and reuse it.
  - For example, if we commonly join multiple tables, we can create a view with the necessary logic. It saves time and reduces repetition.