Day 44 - 90 days of Analytics: Views

In today's first video, we looked at views in SQL

The following were mentioned

-In SQL, a **view** is a virtual table based on the result-set of an SQL statement.

-A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

-We can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

-A view is created with the **CREATE VIEW** statement. Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

-We should note that, a view always shows up-to-date data! The database engine recreates the view, every time a user queries it.

Example

```
CREATE VIEW female_staff AS
SELECT StaffID,Age,Gender
FROM staff_db.staffdemographic
WHERE Gender="Female";
```

-To see the content of the view we created above

```
SELECT *
FROM female_staff;
```

-A view can be updated with the **CREATE OR REPLACE VIEW** statement. Syntax

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example

```
CREATE OR REPLACE VIEW female_staff AS
SELECT StaffID,Age,Gender
FROM staff_db.staffdemographic
WHERE Gender="Female";
```

-A view is deleted with the **DROP VIEW** statement. Syntax

```
DROP VIEW view_name;
```

Example

```
DROP VIEW female_staff;
```

Link to the YouTube Recording: https://www.youtube.com/watch?v=rywMXOROqn0

#90daysofanalytics #community #dataanalysis #dataanalyst #microsoft #msexcel #SQL

Day 44 - 90 days of Analytics: Stored Procedures

In today's second video, we looked at stored procedures with SQL

The following were mentioned

-A **stored procedure** is a prepared SQL code that we can save, so the code can be reused over and over again. So if we have an SQL query that you write over and over again, we can save it as a stored procedure, and then just call it to execute it.

-We can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed. Syntax

```
DELIMITER //
CREATE PROCEDURE procedure_name
BEGIN
sql_statement
END //
```

-To execute a **Stored Procedure** we use the following syntax

```
CALL procedure_name;
```

Example

```
DELIMITER //
CREATE PROCEDURE GetAllMale()
BEGIN
       SELECT *
    FROM staff_db.staffdemographic
    WHERE Gender = "Male";
END //

CALL GetAllMale();
```

-Parameters can be passed to procedures. An example is shown below

```
DELIMITER //
CREATE PROCEDURE GetStaffByJob(IN Job VARCHAR(15))
BEGIN
       SELECT *
    FROM staff_db.staffsalary
    WHERE JobTitle = Job;
END //

CALL GetStaffByJob("Analyst");
```

-A stored procedure can also give produce some output as shown in the example below

```
DELIMITER $$
CREATE PROCEDURE GetCountByJob(IN Job VARCHAR(15), OUT total INT)
BEGIN
       SELECT COUNT(JobTitle) INTO total
    FROM staff_db.staffsalary
    WHERE JobTitle = Job;
END $$

CALL GetCountByJob("Analyst",@total);
SELECT @total AS JobCount;
```

Link to the YouTube Recording: https://www.youtube.com/watch?v=-LzkeGMmjrA

#90daysofanalytics #community #dataanalysis #dataanalyst #microsoft #msexcel #SQL