SQL In Sprints - SQL Starts

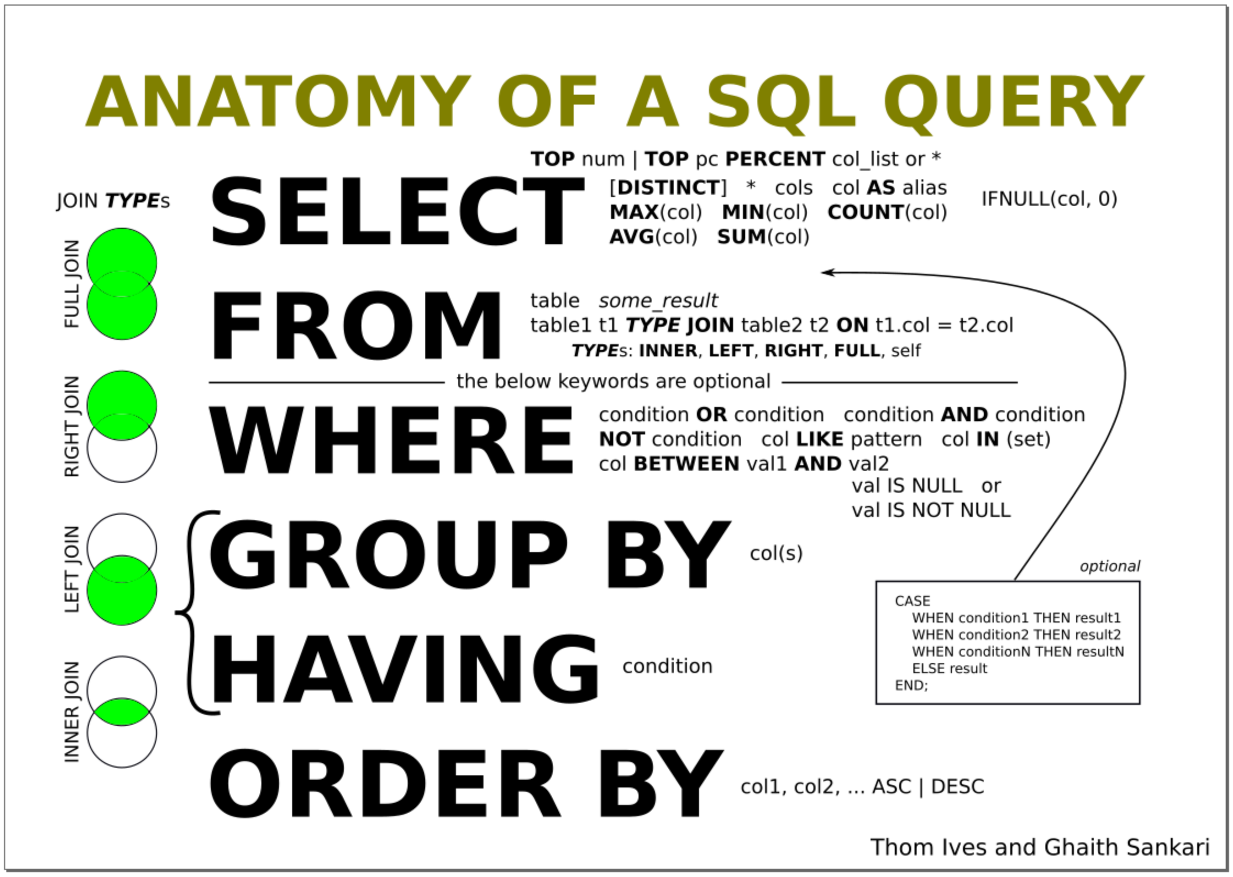

ANATOMY OF A SQL QUERY
JOIN TYPEs
FULL JOIN
RIGHT JOIN
LEFT JOIN
INNER JOIN
SELECT
TOP num | TOP pc PERCENT col_list or *
[DISTINCT] * cols col AS alias
MAX(col) MIN(col) COUNT(col)
AVG(col) SUM(col)
IFNULL(col, 0)
FROM
table some_result
table1 t1 TYPE JOIN table2 t2 ON t1.col = t2.col
TYPEs: INNER, LEFT, RIGHT, FULL, self
the below keywords are optional
WHERE
condition OR condition condition AND condition
NOT condition col LIKE pattern col IN (set)
col BETWEEN val1 AND val2
val IS NULL or
val IS NOT NULL
GROUP BY
col(s)
HAVING
condition
ORDER BY
col1, col2, ... ASC | DESC
optional
CASE
WHEN condition1 THEN result1
WHEN condition2 THEN result2
WHEN conditionN THEN resultN
ELSE result
END;
Thom Ives and Ghaith Sankari

Figure 1: Anatomy Of A SQL Query

Overview

When we first made the above structured cheat sheet, we thought it would be a BIG help. However, we thought it would be a big help for those like Thom. That is people that have learned SQL, but use it infrequently. Due to the ongoing nature of their work, they use it intensely for a time. Then there is a long period of time before their next SQL usage. BUT we had no idea how useful it would become for new SQL users!

A Story

Thom has the current privilege of working with a brilliant Ph.D. candidate that is NOT a data geek. She IS a datavangelist though. What we mean is that she didn't start working with Thom and his team as a data scientist. She comes from the psychological realms of science. However, we greatly value her insights and we love it that she loves data backed agendas. Thus, she needs to know SQL to query data to explore her hypotheses. She reached out to Thom for SQL help.

Normally, Thom might say, "You don't want SQL help from me! I only use it in sprints." But Thom helped develop these cheat sheets. Thom can come out of his SQL dry times fast now with these cheat sheets. So Thom shared these cheat sheets, the one above and the ones below, with said brilliant Ph.D. candidate. Thom was thinking, 'She'll take a look at these, and then we can talk through how to structure her query'.

Before Thom could wrap up the current task he was working on, said brilliant Ph.D. candidate (BPHDC) sent him a note.

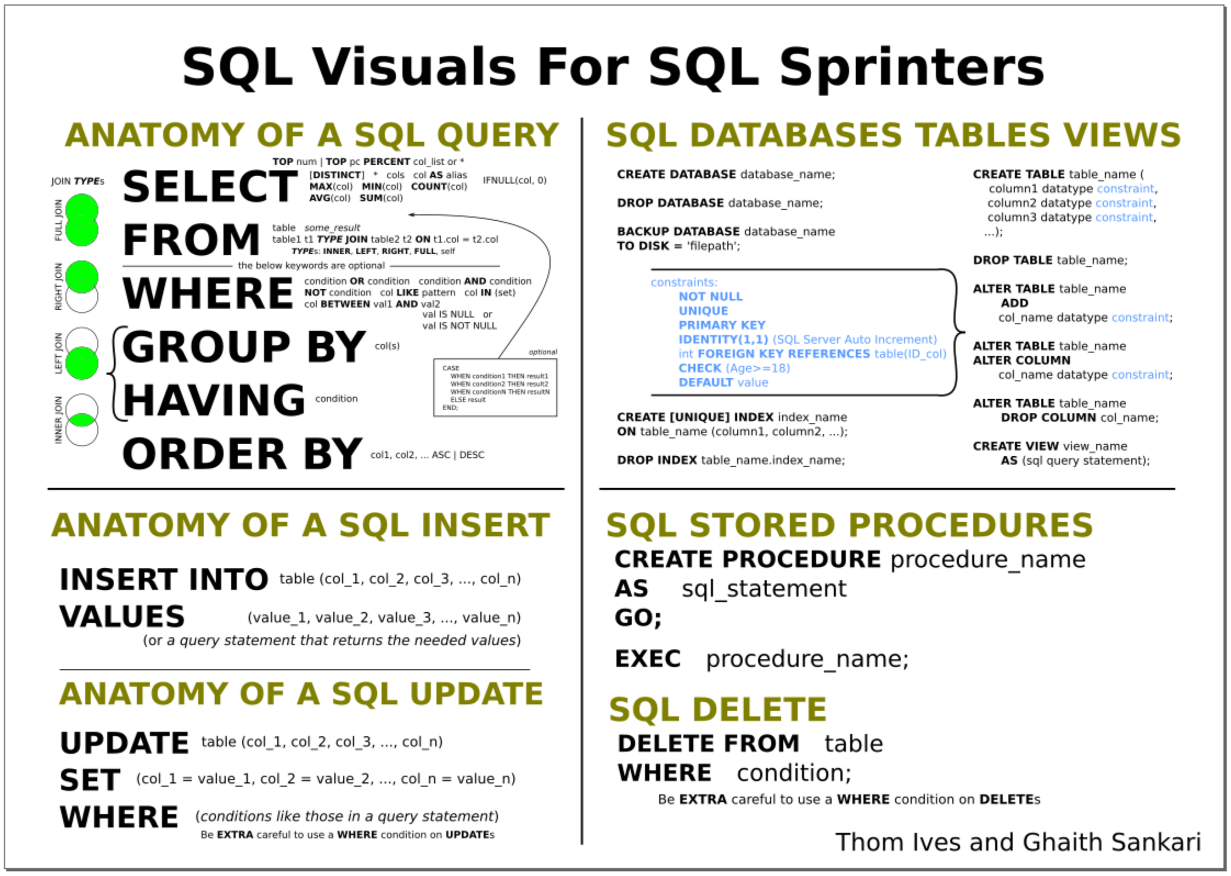
BPHDC: “Thanks Thom! I got it” Thom: “Wait! Got What? You mean you got the cheat sheets?” BPHDC: “Well, yes I got those, but I meant that I have my query working now. I even added that join that I needed.” Thom: “What?!?! Just from looking at those cheat sheets?” BPHDC: “Yes. Thanks. That helped to clear up my confusion on how to structure my query.”

NOTE*: Thom cleaned up some language, and this is from memory, but this was how things went.*

This was not something we expected from the value of these cheat sheets, BUT it kinda makes sense too. SQL is very logical in it’s structure, but the **ORDER** that key commands should come in and the options you can and should use can be hard to remember sometimes. BPHDC has a brilliant mind and already thinks logically, but we all need syntax and keyword reminders. That’s all BPHDC needed to SPRINT with SQL. So, we want to get these sheets in the hands and minds of as many people as possible to help them too.

The Cheat Sheets

All Four Cheat Sheets In One Image



SQL Visuals For SQL Sprinters

ANATOMY OF A SQL QUERY

JOIN *TYPE*s

FULL JOIN

RIGHT JOIN

LEFT JOIN

INNER JOIN

SELECT **TOP** num | **TOP** pc **PERCENT** col_list or *
[**DISTINCT**] * cols col **AS** alias IFNULL(col, 0)
MAX(col) **MIN**(col) **COUNT**(col)
AVG(col) **SUM**(col)

FROM table *some_result*
table1 t1 ***TYPE* JOIN** table2 t2 **ON** t1.col = t2.col
***TYPE*s: INNER, LEFT, RIGHT, FULL,** self

the below keywords are optional

WHERE condition **OR** condition condition **AND** condition
NOT condition col **LIKE** pattern col **IN** (set)
col **BETWEEN** val1 **AND** val2
val IS NULL or
val IS NOT NULL

GROUP BY col(s)

HAVING condition

optional

```
CASE
   WHEN condition1 THEN result1
   WHEN condition2 THEN result2
   WHEN conditionN THEN resultN
   ELSE result
END;
```

ORDER BY col1, col2, ... ASC | DESC

SQL DATABASES TABLES VIEWS

CREATE DATABASE database_name;

DROP DATABASE database_name;

BACKUP DATABASE database_name
TO DISK = 'filepath';

constraints:
NOT NULL
UNIQUE
PRIMARY KEY
IDENTITY(1,1) (SQL Server Auto Increment)
int **FOREIGN KEY REFERENCES** table(ID_col)
CHECK (Age>=18)
DEFAULT value

CREATE [UNIQUE] INDEX index_name
ON table_name (column1, column2, ...);

DROP INDEX table_name.index_name;

CREATE TABLE table_name (
column1 datatype constraint,
column2 datatype constraint,
column3 datatype constraint,
...);

DROP TABLE table_name;

ALTER TABLE table_name
ADD
col_name datatype constraint;

ALTER TABLE table_name
ALTER COLUMN
col_name datatype constraint;

ALTER TABLE table_name
DROP COLUMN col_name;

CREATE VIEW view_name
AS (sql query statement);

ANATOMY OF A SQL INSERT

INSERT INTO table (col_1, col_2, col_3, ..., col_n)
VALUES (value_1, value_2, value_3, ..., value_n)
(or *a query statement* that returns the needed values)

ANATOMY OF A SQL UPDATE

UPDATE table (col_1, col_2, col_3, ..., col_n)
SET (col_1 = value_1, col_2 = value_2, ..., col_n = value_n)
WHERE (*conditions like those in a query statement*)
Be **EXTRA** careful to use a **WHERE** condition on **UPDATE**s

SQL STORED PROCEDURES

CREATE PROCEDURE procedure_name
AS sql_statement
GO;

EXEC procedure_name;

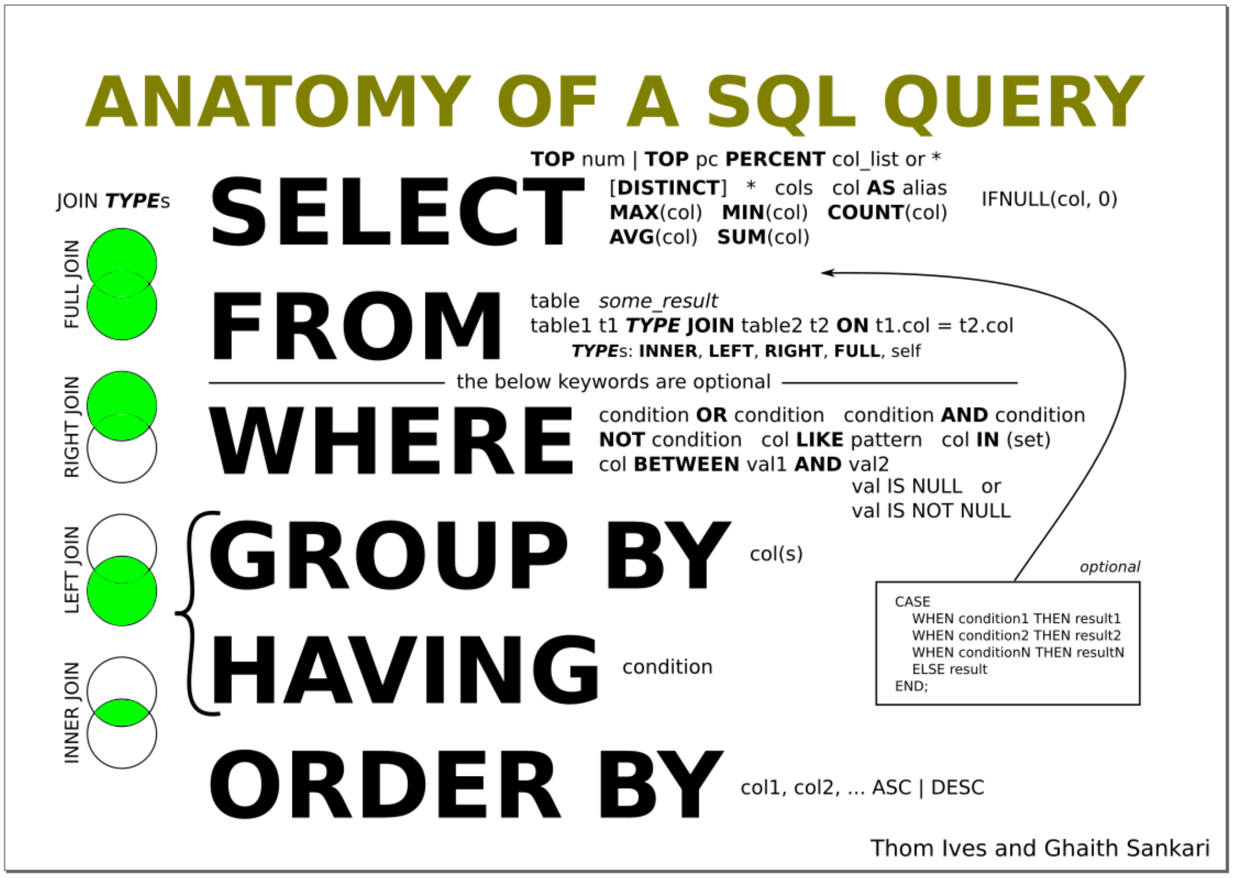
SQL DELETE

DELETE FROM table
WHERE condition;
Be **EXTRA** careful to use a **WHERE** condition on **DELETE**s

Thom Ives and Ghaith Sankari

Figure 2: SQL Operations Summary

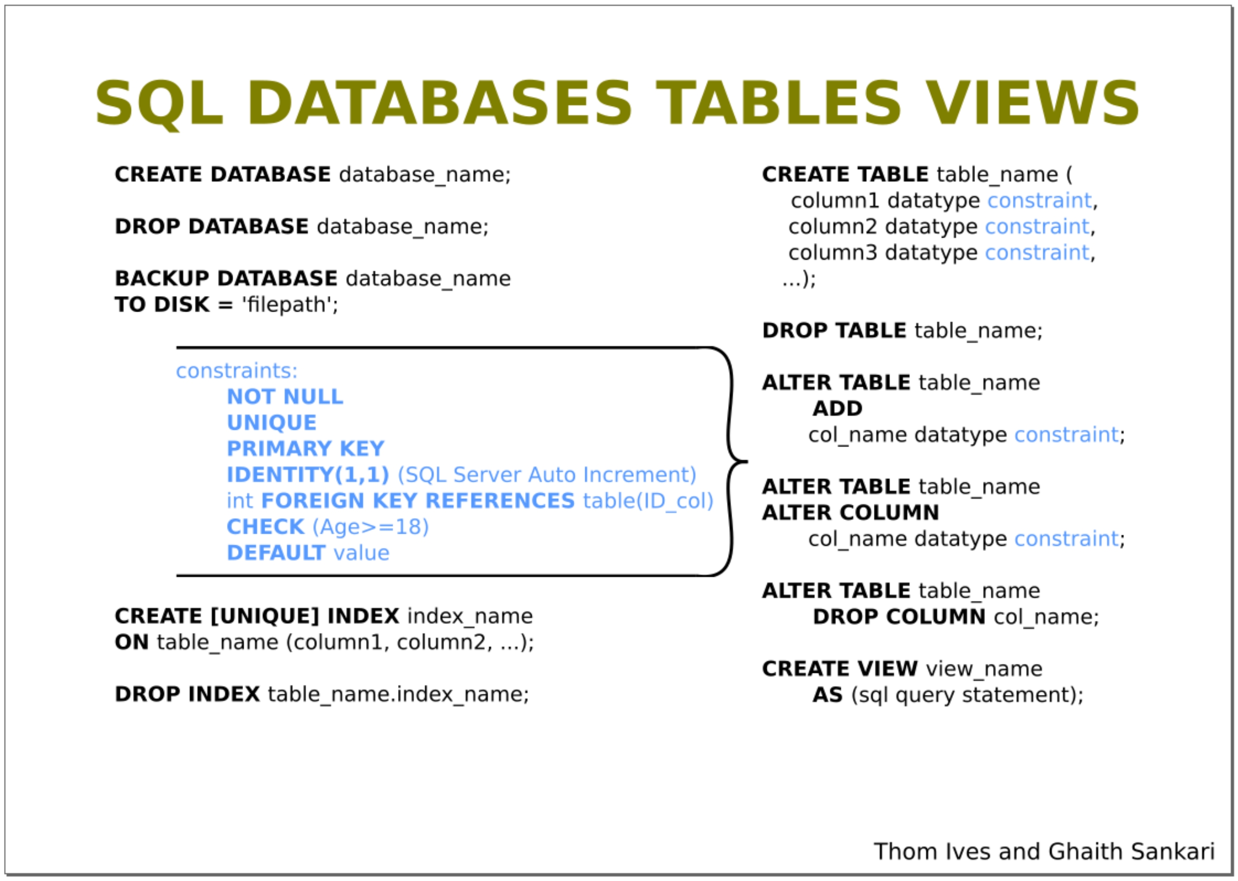
The Anatomy Of A SQL Query Repeated


ANATOMY OF A SQL QUERY
JOIN TYPEs
FULL JOIN
RIGHT JOIN
LEFT JOIN
INNER JOIN
SELECT
TOP num | TOP pc PERCENT col_list or *
[DISTINCT] * cols col AS alias
MAX(col) MIN(col) COUNT(col)
AVG(col) SUM(col)
IFNULL(col, 0)
FROM
table some_result
table1 t1 TYPE JOIN table2 t2 ON t1.col = t2.col
TYPEs: INNER, LEFT, RIGHT, FULL, self
the below keywords are optional
WHERE
condition OR condition condition AND condition
NOT condition col LIKE pattern col IN (set)
col BETWEEN val1 AND val2
val IS NULL or
val IS NOT NULL
GROUP BY
col(s)
HAVING
condition
ORDER BY
col1, col2, ... ASC | DESC
optional
Thom Ives and Ghaith Sankari

```
CASE
   WHEN condition1 THEN result1
   WHEN condition2 THEN result2
   WHEN conditionN THEN resultN
   ELSE result
END;
```

Figure 3: Anatomy Of A SQL Query

Creating SQL Databases, Tables, and Views



SQL DATABASES TABLES VIEWS

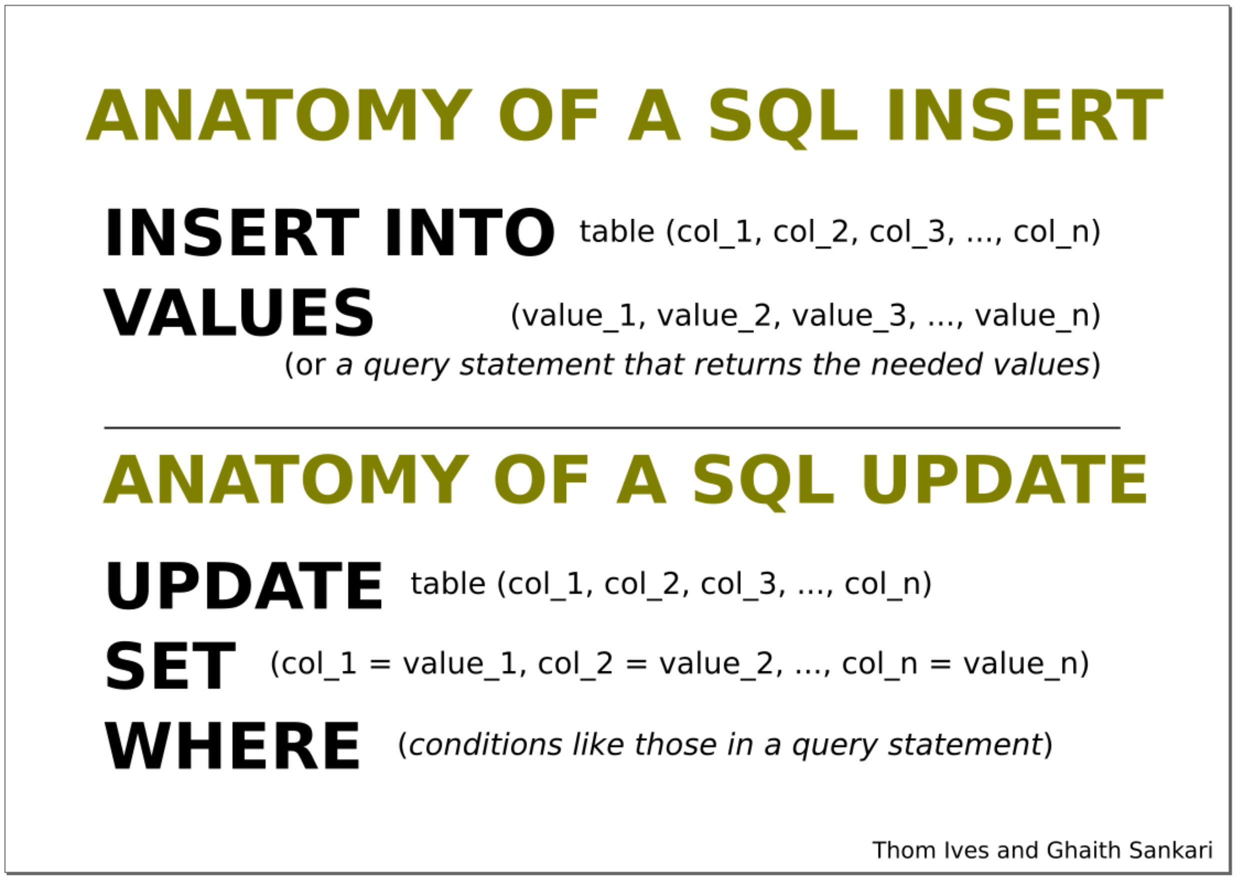
```
CREATE DATABASE database_name;

DROP DATABASE database_name;

BACKUP DATABASE database_name
TO DISK = 'filepath';

    constraints:
        NOT NULL
        UNIQUE
        PRIMARY KEY
        IDENTITY(1,1) (SQL Server Auto Increment)
        int FOREIGN KEY REFERENCES table(ID_col)
        CHECK (Age>=18)
        DEFAULT value

CREATE [UNIQUE] INDEX index_name
ON table_name (column1, column2, ...);

DROP INDEX table_name.index_name;
```

```
CREATE TABLE table_name (
   column1 datatype constraint,
   column2 datatype constraint,
   column3 datatype constraint,
  ...);

DROP TABLE table_name;

ALTER TABLE table_name
    ADD
    col_name datatype constraint;

ALTER TABLE table_name
ALTER COLUMN
    col_name datatype constraint;

ALTER TABLE table_name
    DROP COLUMN col_name;

CREATE VIEW view_name
    AS (sql query statement);
```

Thom Ives and Ghaith Sankari

Figure 4: Creating SQL Databases, Tables, and Views

SQL Inserts And Updates



ANATOMY OF A SQL INSERT

INSERT INTO table (col_1, col_2, col_3, ..., col_n)

VALUES (value_1, value_2, value_3, ..., value_n)

(or *a query statement that returns the needed values*)

ANATOMY OF A SQL UPDATE

UPDATE table (col_1, col_2, col_3, ..., col_n)

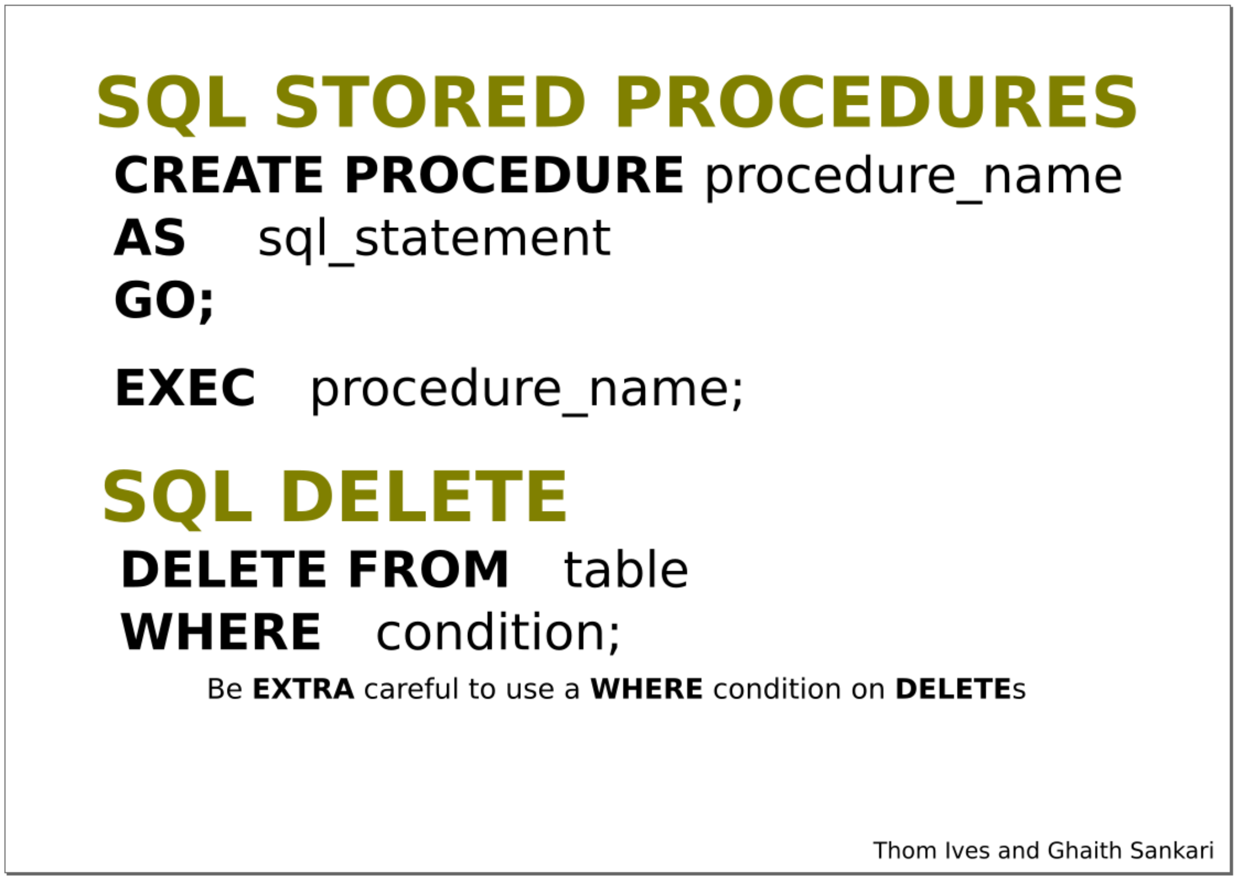
SET (col_1 = value_1, col_2 = value_2, ..., col_n = value_n)

WHERE (*conditions like those in a query statement*)

Thom Ives and Ghaith Sankari

Figure 5: SQL Inserts And Updates

SQL Stored Procedures And Deletes



SQL STORED PROCEDURES

```
CREATE PROCEDURE procedure_name
AS    sql_statement
GO;

EXEC   procedure_name;
```

SQL DELETE

```
DELETE FROM   table
WHERE   condition;
```

Be **EXTRA** careful to use a **WHERE** condition on **DELETE**s

Thom Ives and Ghaith Sankari

Figure 6: SQL Stored Procedures

Summary

SQL is important to ALL that love and need data for their work. Please save this and share it with your friends around the world. It's our hope that it will help MANY more people. And Ghaith and Thom would be honored if you would follow us on LinkedIn. We have many more ways that we want to help the data world. We are developing the best way possible to deliver that help.

https://www.linkedin.com/in/thomives/ https://www.linkedin.com/in/ghaith-sankari-a3449a2a/