Complete SQL Syntax Cheat Sheet: PostgreSQL vs SQL Server

LIMIT Rows

```
PostgreSQL:
SELECT * FROM table LIMIT 10;
SQL Server:
SELECT TOP 10 * FROM table;
```

OFFSET + LIMIT

```
PostgreSQL:
SELECT * FROM table ORDER BY id OFFSET 20 LIMIT 10;
SQL Server:
SELECT * FROM table ORDER BY id OFFSET 20 ROWS FETCH NEXT 10 ROWS ONLY;
```

String Concatenation

```
PostgreSQL:
SELECT 'A' || 'B';
SQL Server:
SELECT 'A' + 'B';
```

Substring

```
PostgreSQL:
SUBSTRING(col FROM 1 FOR 3)
SQL Server:
SUBSTRING(col, 1, 3)
```

String Position

```
PostgreSQL:
POSITION('x' IN col)
SQL Server:
CHARINDEX('x', col)
```

String Length

```
PostgreSQL:
LENGTH(col)
SQL Server:
LEN(col)
```

Current Timestamp

```
PostgreSQL:
NOW()
SQL Server:
GETDATE()
```

Extract Date Part

```
PostgreSQL:
EXTRACT(YEAR FROM date_col)
```

```
SQL Server:
YEAR(date_col)
```

Add Interval

```
PostgreSQL:
date_col + INTERVAL '5 days'
SQL Server:
DATEADD(DAY, 5, date_col)
```

Date Difference

```
PostgreSQL:
date2 - date1
SQL Server:
DATEDIFF(DAY, date1, date2)
```

Natural Log

```
PostgreSQL:
LN(val)
SQL Server:
LOG(val)
```

Log Base 10

```
PostgreSQL:
LOG(val)
SQL Server:
LOG10(val)
```

Variable Declaration

```
PostgreSQL:
DO $$ DECLARE x INT := 5; BEGIN RAISE NOTICE 'x = %', x; END $$;
SQL Server:
DECLARE @x INT = 5; PRINT @x;
```

UPSERT

```
PostgreSQL:
INSERT INTO t (id, val) VALUES (1, 'a') ON CONFLICT (id) DO UPDATE SET val = EXCLUDED.val;
SQL Server:
MERGE t AS tgt USING (SELECT 1 AS id, 'a' AS val) AS src ON tgt.id = src.id WHEN MATCHED THEN
UPDATE SET tgt.val = src.val WHEN NOT MATCHED THEN INSERT (id, val) VALUES (src.id, src.val);
```

Temp Table

```
PostgreSQL:
CREATE TEMP TABLE temp (col INT);
SQL Server:
CREATE TABLE #temp (col INT);
```

Complete SQL Syntax Cheat Sheet: PostgreSQL vs SQL Server

Auto Increment

```
PostgreSQL:
id SERIAL PRIMARY KEY
SQL Server:
id INT IDENTITY(1,1) PRIMARY KEY
```

Boolean Type

```
PostgreSQL:
is_valid BOOLEAN
SQL Server:
is_valid BIT
```

Arrays

```
PostgreSQL:
tags TEXT[]
SQL Server:
Not Supported
```

Note: Use normalized tables instead

JSON

```
PostgreSQL:
col JSON / JSONB
SQL Server:
col NVARCHAR(MAX); Use OPENJSON()
```

IF Statement

```
PostgreSQL:
IF cond THEN ... END IF;
SQL Server:
IF cond BEGIN ... END
```

Exception Handling

```
PostgreSQL:
BEGIN ... EXCEPTION WHEN ... THEN ... END;
SQL Server:
BEGIN TRY ... END TRY BEGIN CATCH ... END CATCH
```

Function Creation

```
PostgreSQL:
CREATE FUNCTION fn(...) RETURNS INT AS $$ BEGIN ... END $$ LANGUAGE plpgsql;
SQL Server:
CREATE FUNCTION fn(...) RETURNS INT AS BEGIN ... END
```

CTE

```
PostgreSQL:
```

Complete SQL Syntax Cheat Sheet: PostgreSQL vs SQL Server

```
WITH RECURSIVE cte AS (...) SELECT * FROM cte;
SQL Server:
WITH cte AS (...) SELECT * FROM cte OPTION (MAXRECURSION 100);
```

Index Types

```
PostgreSQL:
GIN, GiST, BRIN
SQL Server:
Only B-Tree; full-text via separate service
```

Default Schema

```
PostgreSQL:
`public`
SQL Server:
`dbo`
```

Permissions

```
PostgreSQL:
GRANT USAGE ON SCHEMA ...
SQL Server:
GRANT SELECT, INSERT ON ... TO ...;
```

Note: PostgreSQL roles are more Unix-like