Intro to SQL Server - John MacKintosh

Note as of Dec 2019
Note Taker: Paris Zhang

Ch 1 - SELECT and WHERE **1**

Ch 2 - Groups, strings, and counting things **1**
String Manipulation: RIGHT, LEN, CHARINDEX and SUBSTRING 1
GROUPING and HAVING 2

Ch 3 - Joining tables **2**

Ch 4 - Creating Tables **3**
CRUD operations 3
Data Types (Common) 4
INSERT, UPDATE, and DELETE 4
DECLARE 5
DECLARE and SET 5
Temporary Table 5

Course Summary **6**

Ch 1 - SELECT and WHERE

SELECT `TOP (X) PERCENT` to select the top X% of rows
e.g., SELECT TOP(5) PERCENT artist

Ch 2 - Groups, strings, and counting things

String Manipulation: `RIGHT`, `LEN`, `CHARINDEX` and `SUBSTRING`

SELECT `LEN(X)` to return the length of a string, `LEFT(X, n)` or `RIGHT(X, n)` to select the first/last n characters of the string X
e.g., RIGHT(description, 20) AS last_20

SELECT `CHARINDEX ('x' , X)` to find the index (location) of the character 'x' in the string X

```
SELECT
  CHARINDEX ('_', url) AS char_location,
  url
FROM courses;
```

```
+---------------+-------------------------------------+
| char_location | url                                 |
|---------------+-------------------------------------|
| 34            | datacamp.com/courses/introduction_  |
| 34            | datacamp.com/courses/intermediate_  |
| 29            | datacamp.com/courses/writing_       |
| 29            | datacamp.com/courses/joining_       |
| 27            | datacamp.com/courses/intro_         |
+---------------+-------------------------------------+
```

SELECT `SUBSTRING(X, index_start, length_to_extract)` to extract a portion of a string X

```
SELECT
  SUBSTRING(url, 12, 12) AS target_section,
  url
FROM courses;
```

```
+-----------------+--------------------------------+
| target_section  | url                            |
+-----------------+--------------------------------+
| www.datacamp.com |https//www.datacamp.com/courses |
+-----------------+--------------------------------+
```

SELECT `REPLACE(X, replace_from , replace_to)` to replace a character in a string

```
SELECT
  TOP(5) REPLACE(url, '_', '-') AS replace_with_hyphen
FROM courses;
```

```
+--------------------------------------+
| replace_with_hyphen                  |
+--------------------------------------|
| datacamp.com/courses/introduction-   |
| datacamp.com/courses/intermediate-   |
| datacamp.com/courses/writing-        |
| datacamp.com/courses/joining-        |
| datacamp.com/courses/intro-          |
+--------------------------------------+
```

GROUPING and HAVING

Summary

- GROUP BY splits the data up into combinations of one or more values
- WHERE filters on row values (appears before GROUP BY)
- HAVING appears after the GROUP BY clause and filters on groups or aggregates

```
SELECT
  SUM(demand_loss_mw) AS lost_demand,
  description
FROM grid
WHERE
  description LIKE '%storm'
  AND demand_loss_mw IS NOT NULL
GROUP BY description
HAVING SUM(demand_loss_mw) > 1000;
```

```
+----------- -+-----------------------------------------+
| lost_demand | description                             |
|-------------+-----------------------------------------|
| 4171        | Severe Weather  Winter Storm            |
| 1352        | Winter Storm                            |
+-------------+-----------------------------------------+
```

Ch 3 - Joining tables

`INNER JOIN`, `LEFT JOIN`, and `RIGHT JOIN`

`UNION` excludes duplicate rows, whereas `UNION ALL` captures all rows.

UNION	UNION ALL

```sql
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 3)
UNION
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 4, 5);
```

```sql
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 3)
UNION ALL
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 4, 5);
```

```
+------------+------------------------+------------+
| album_id   | title                  | artist_id  |
|------------+------------------------|------------|
| 1          | For Those About To Rock | 1          |
| 4          | Let There Be Rock      | 1          |
| 5          | Big Ones               | 3          |
| 6          | Jagged Little Pill     | 4          |
| 7          | Facelift               | 5          |
+------------+------------------------+------------+
```

```
+------------+------------------------+------------+
| album_id   | title                  | artist_id  |
|------------+------------------------|------------|
| 1          | For Those About To Rock | 1          |
| 4          | Let There Be Rock      | 1          |
| 5          | Big Ones               | 3          |
| 1          | For Those About To Rock | 1          |
| 4          | Let There Be Rock      | 1          |
| 6          | Jagged Little Pill     | 4          |
| 7          | Facelift               | 5          |
+------------+------------------------+------------+
```

Summary

1. `UNION` or `UNION ALL`: Combines queries from the same table or different tables
    a. If combining data from different tables:
    - Select the same number of columns in the same order
    - Columns should have the same data types
    b. If source tables have different column names
    - Alias the column names
2. UNION: Discards duplicates (slower to run)
3. UNION ALL : Includes duplicates (faster to run)

Ch 4 - Creating Tables

CRUD operations

CREATE

- Databases, Tables or views
- Users, permissions, and security groups

READ

- Example: SELECT statements

UPDATE

- Amend existing database records

DELETE

CREATE	`CREATE TABLE` `unique table name` (column name, data type, size)	`CREATE TABLE test_table( test_date date, test_name varchar(20), test_int int )`

Data Types (Common)

Dates:

- date (YYYY-MM-DD), datetime (YYYY-MM-DD hh:mm:ss)
- time

Numeric:

- integer, decimal, oat
- bit (1 = TRUE , 0 = FALSE. Also accepts NULL values)

Strings:

- char , varchar , nvarchar

INSERT, UPDATE, and DELETE

INSERT	`INSERT INTO table_name (col1, col2, col3) VALUES ('value1', 'value2', value3)`	
INSERT SELECT	`INSERT INTO table_name (col1, col2, col3) SELECT column1, column2, column3 FROM other_table WHERE -- conditions apply`	- Don't use SELECT * - Be specific in case table structure changes

UPDATE	`UPDATE table`
	`SET`
	`  column1 = value1,`
	`  column2 = value2`
	`WHERE`
	`  -- Condition(s);`
		Remember the `WHERE` clause

DELETE	`DELETE`
	`FROM table`
	`WHERE`
	`  -- Conditions` or `TRUNCATE TABLE table_name`	

DECLARE

DECLARE and SET

```
DECLARE @my_artist varchar(100)
DECLARE @my_album varchar(300);

SET @my_artist = 'AC/DC'
SET @my_album  = 'Let There Be Rock' ;

SELECT --
FROM --
WHERE artist = @my_artist
AND album = @my_album;
```

```
DECLARE @my_artist varchar(100)
DECLARE @my_album varchar(300);

SET @my_artist = 'U2'
SET @my_album  = 'Pop' ;

SELECT --
FROM --
WHERE artist = @my_artist
AND album = @my_album;
```

Temporary Table

Using `INTO`

```
SELECT
  col1,
  col2,
  col3 INTO #my_temp_table
FROM my_existing_table
WHERE
  -- Conditions
```

- `#my_temp_table` exists until connection or session ends

```
-- Remove table manually
DROP TABLE #my_temp_table
```

Course Summary

Selecting: `SELECT`

Ordering: `ORDER BY`

Filtering: `WHERE` and `HAVING`

Aggregating: `SUM`, `COUNT`, `MIN`, `MAX` and `AVG`

Text manipulation: `LEFT`, `RIGHT`, `LEN` and `SUBSTRING`

`GROUP BY`

`INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`

`UNION` and `UNION ALL`

Create, Read, Update and Delete (CRUD)

Variables

Temporary tables