

Rotman

**Master of
Management
Analytics**

INTRO TO SQL

Bootcamp (<https://tdmdal.github.io/mma-sql-2021/>)

July 22, 2021 Prepared by Jay / [TDMDAL](#)



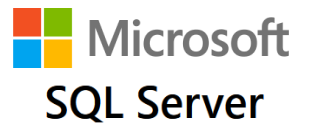
Rotman School of Management
UNIVERSITY OF TORONTO

What's SQL (Structured Query Language)




- Most widely used database (DB) language
 - a domain specific language (managing data stored in relational DB)
- Not a proprietary language
 - Open specifications/standards
 - All major DBMS (DB Mgmt. System) vendors implement ANSI Standard SQL
 - However, SQL Extensions are usually DB specific
- Powerful despite simplicity

What's DB and DB Management System

- What's a database: A collection of data in an organized way
- Relational DB
 - tables
 - columns/fields/variables and datatypes
 - rows/records/observations
 - primary key, foreign key, constraints and relationships (discuss later)
 - other objects: indices, views, etc.
- What is DBMS (DB Management System)?
 - A software system that manages/maintains relational DBs
 - e.g., MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, etc.



Connect to a DB and use SQL – DB Client

- DB specific management client
 - command-line console
 - GUI (Graphic User Interface) client
 - e.g., [DB Browser for SQLite](#), [MySQL Workbench](#), [pgAdmin for PostgreSQL](#), [MS SSMS](#)
- Generic DB client can connect to different DBs through connectors
 - GUI client (e.g. [DBeaver](#), [Beekeeper Studio](#), [Navicat](#))   
 - Programming language
 - e.g., Python + [SQLAlchemy](#) + DBAPI (e.g. [SQLite](#), [MySQL](#), [PostgreSQL](#), etc.), R + [dbplyr](#)
 - In this workshop: Python + [ipython-sql notebook magic](#)

Beyond a relational DB language

- SAS's PROC SQL
- Spark's SparkSQL
 - [Apache Spark](#) is a big data computing framework
- Hive's HiveQL, an SQL-like query language
 - [Apache Hive](#) is a distributed data warehouse (data warehouse?)
- Google BigQuery's SQL
 - BigQuery is Google's data warehouse (analyze petabytes of data at ease)

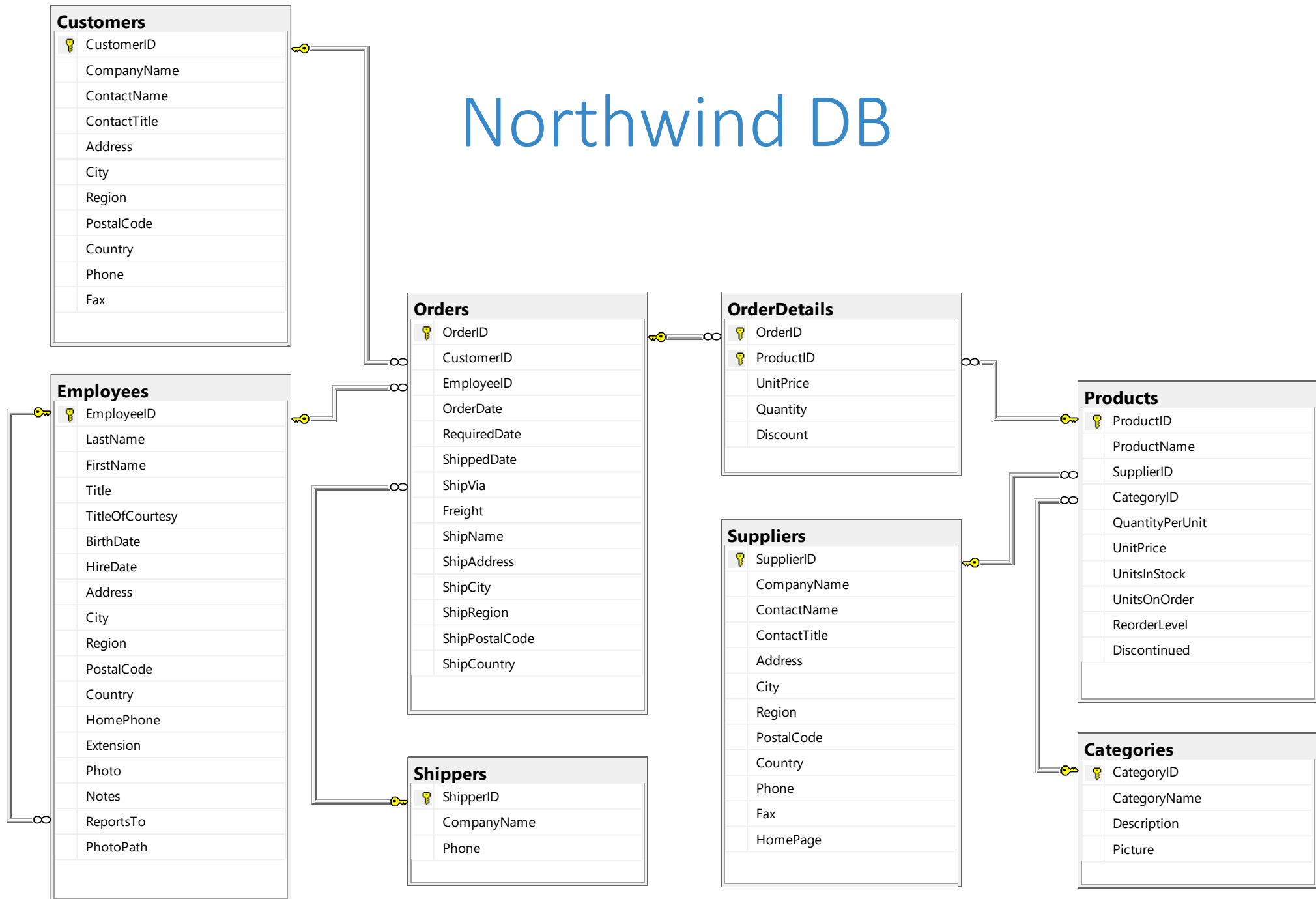
SQL Hands-on Exercises (Learning-by-doing)

- Course website: <https://tdmdal.github.io/mma-sql-2021/>
- Google Colab
 - Google's Jupyter Notebook
 - A notebook can contain live code, equations, visualizations and narrative text
- Why SQLite?
 - a [small](#), [fast](#), [self-contained](#), [high-reliability](#), [full-featured](#), SQL DB engine
 - perfect for learning SQL

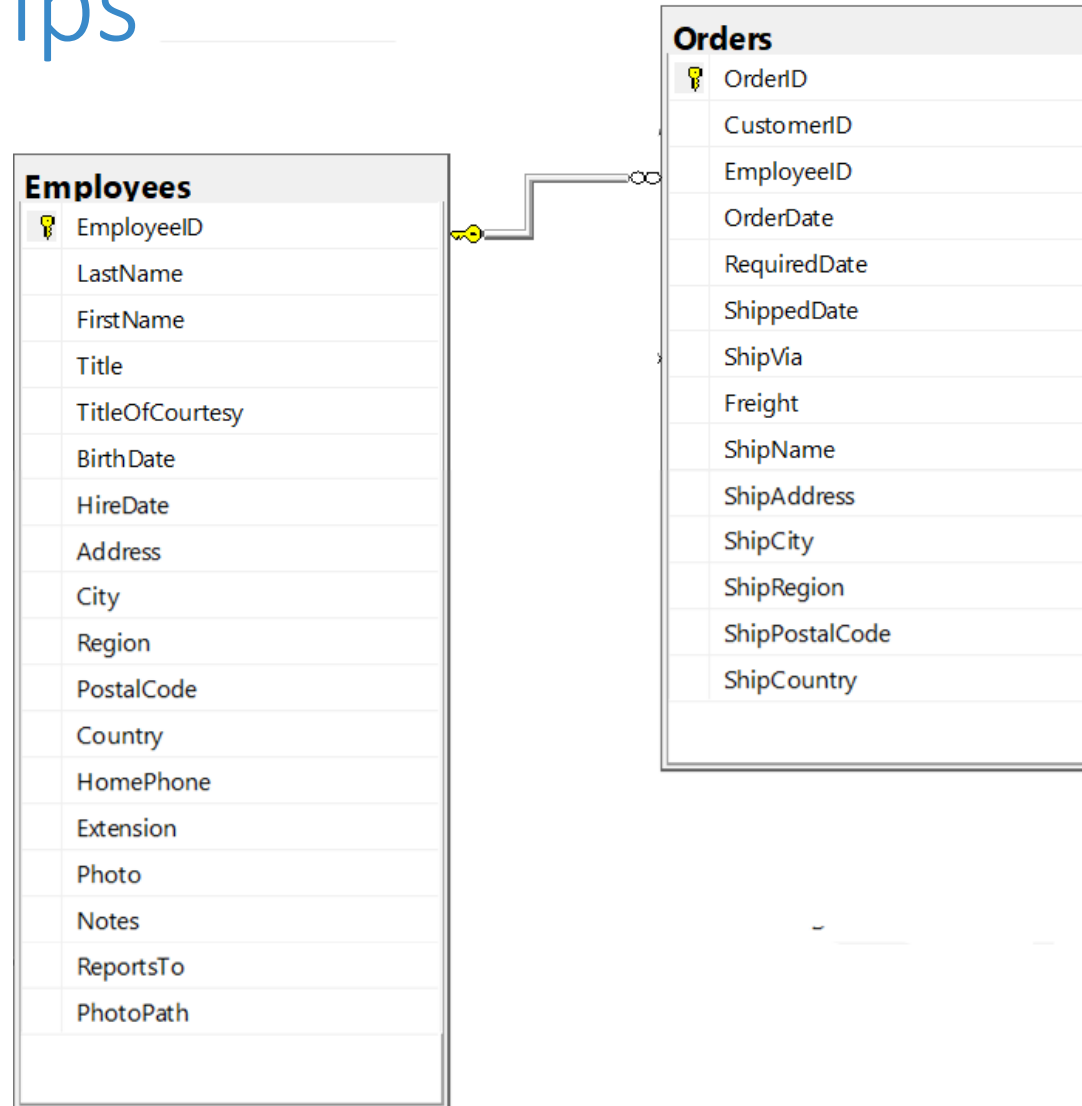
Preparation For RSM8411 (MMA, Fall 2021)

- A different setup (a more advanced/powerful DBMS)
 - [PostgreSQL](#), an open-source DBMS
 - [pgAdmin](#), a GUI client for managing PostgreSQL
 - Installation guide and get-started resources: see our [bootcamp website](#)
- Please make sure you have the above setup installed
 - Set it up before the end of this bootcamp
 - Email me if you have trouble with installation
- SQL syntax difference between SQLite and PostgreSQL
 - For 99% of what we will learn in this bootcamp, they are the same

Northwind DB



Primary key, foreign key, constraints and relationships



Hands-on Part 1: Warm up

- Retrieve data: `SELECT . . . FROM . . .`
- Sort retrieved data: `SELECT . . . FROM . . . ORDER BY . . .`
- Filter data: `SELECT . . . FROM . . . WHERE . . .`
 - `IN`, `NOT`, `LIKE` and `%` wildcard
- Create calculated fields
 - mathematical calculations (e.g. `+`, `-`, `*`, `/`)
 - data manipulation functions (e.g. `DATE()`, `||`)

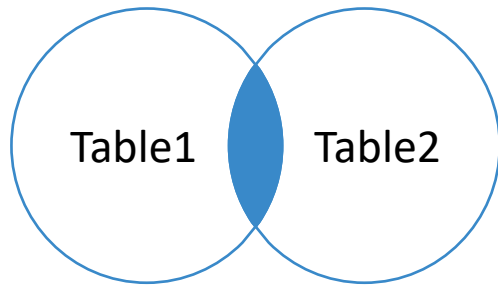
Hands-on Part 2: Summarize and Group Data

- Summarize data using aggregate functions (e.g. `COUNT()`, `MIN()`, `MAX()`, and `AVG()`).
- Group data and filter groups: `SELECT . . . FROM . . . GROUP BY . . . HAVING . . .`
- SELECT clause ordering: `SELECT . . . FROM . . . WHERE . . . GROUP BY . . . HAVING . . . ORDER BY . . .`
- Filter data by subquery:
`SELECT . . . FROM . . . WHERE . . . (SELECT . . . FROM . . .)`

Hands-on Part 2: Join Tables

- Inner join: `SELECT...FROM...INNER JOIN...ON...`
- Left join: `SELECT...FROM...LEFT JOIN...ON...`
- Other join variations.

Join – Inner Join



```
SELECT *  
FROM Table1  
    INNER JOIN Table2  
    ON Table1.pk = Table2.fk;
```

Table1

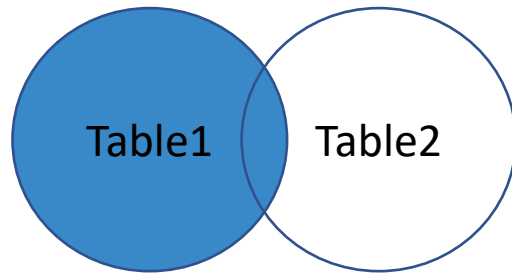
pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d

Join – Left (Outer) Join



```
SELECT *  
FROM Table1  
  LEFT JOIN Table2  
    ON Table1.pk = Table2.fk;
```

Table1

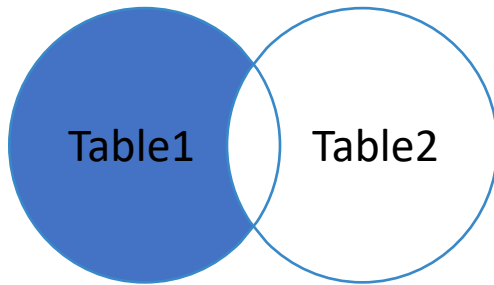
pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d
2	b	null	null

Join - Left (Outer) Join With Exclusion



```
SELECT *  
FROM Table1  
  LEFT JOIN Table2  
    ON Table1.pk = Table2.fk  
WHERE Table2.fk is NULL;
```

Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
2	b	null	null

Join – Right Outer Join*

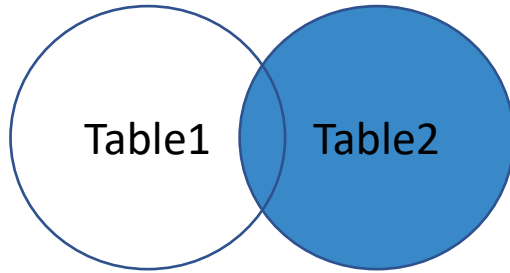


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

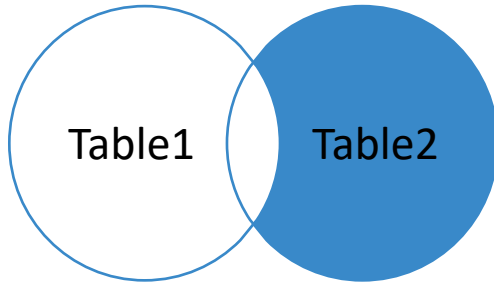
```
SELECT *  
FROM Table2  
  LEFT JOIN Table1  
    ON Table2.fk = Table1.pk
```

```
-----  
SELECT *  
FROM Table1  
  RIGHT JOIN Table2  
    ON Table1.pk = Table2.fk;
```

} SQLite doesn't support this RIGHT JOIN key word, but some DBMSs do (e.g. MySQL).

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d
null	null	3	e

Join - Right Outer Join With Exclusion*



```
SELECT *  
FROM Table2  
  LEFT JOIN Table1  
    ON Table2.fk = Table1.pk  
WHERE Table1.pk is NULL;
```

```
-----  
SELECT *  
FROM Table1  
  RIGHT JOIN Table2  
    ON Table1.pk = Table2.fk  
WHERE Table1.pk is NULL;
```

Table1

pk	t1c1
1	a
2	b

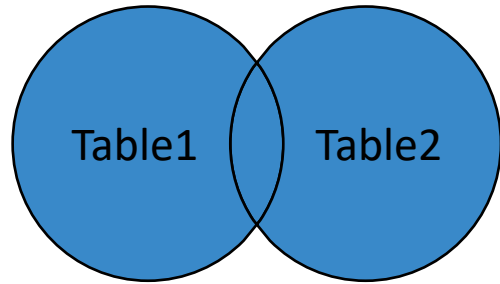
Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
null	null	3	e

} SQLite doesn't support this RIGHT JOIN key word, but some DBMSs do (e.g. MySQL).

Join – Full Outer Join



```
SELECT pk, t1c1, fk, t2c1
FROM Table1
  LEFT JOIN Table2
    ON Table1.pk = Table2.fk
UNION
SELECT pk, t1c1, fk, t2c1
FROM Table2
  LEFT JOIN Table1
    ON Table2.fk = Table1.pk;
```

Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d
2	b	null	null
null	null	3	e

Note: Some DBMS support FULL OUTER JOIN keyword (e.g. MS SQL) so you don't need to do it the above way.

Join – Full Outer Join With Exclusion*

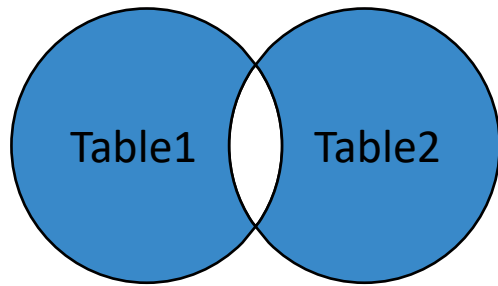


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

```
SELECT pk, t1c1, fk, t2c1
FROM Table1
  LEFT JOIN Table2
    ON Table1.pk = Table2.fk
WHERE Table2.fk is NULL
UNION
SELECT pk, t1c1, fk, t2c1
FROM Table2
  LEFT JOIN Table1
    ON Table2.fk = Table1.pk
WHERE Table1.pk is NULL;
```

pk	t1c1	fk	t2c1
2	b	null	null
null	null	3	e

Others

- CTE and temporary table
- Self-join
- CASE keyword
- UNION keyword

Many things we didn't cover

- Insert data (`INSERT INTO...VALUES...; INSERT INTO...SELECT...FROM...`)
- Update data (`UPDATE...SET...WHERE...`)
- Delete data (`DELETE FROM...WHERE...`)
- Manipulate tables (`CREATE TABLE...; ALTER TABLE...; DROP TABLE...`)
- Views (`CREATE VIEW...AS...`)

The list goes on and on

- Stored procedures
- Functions
- Transaction processing
- Cursors (going through table row by row)
- WINDOW function
- Query optimization
- DB permissions & security
- ...

Ref. A stack overflow discussion on [What is “Advanced” SQL.](#)