Northeastern University

DS 5110 – Lecture 5
SQL Part I

Roi Yehoshua

Agenda

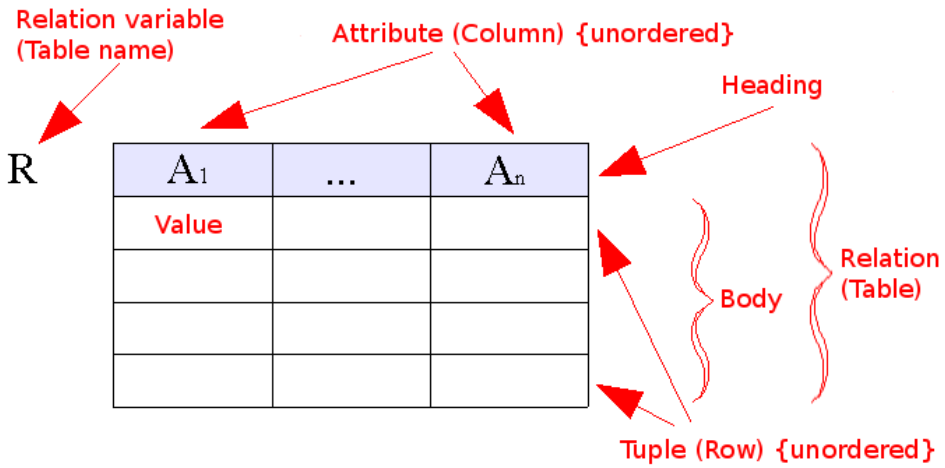
Northeastern University

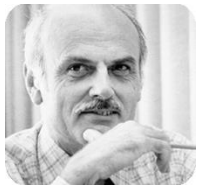
- Relational databases
- The SQL language
- DDL commands
- DML commands
- Select commands
- Aggregate functions
- Grouping
- Join

The Relational Model

Northeastern University

- The most widely used data model
- Introduced by Edgar F. Codd at IBM in 1970
- Data is represented by a collection of **tables** (also called **relations**)
- Each table contains **records** (**rows**) of a particular type
- Each record type defines a fixed number of **fields** (**columns**)



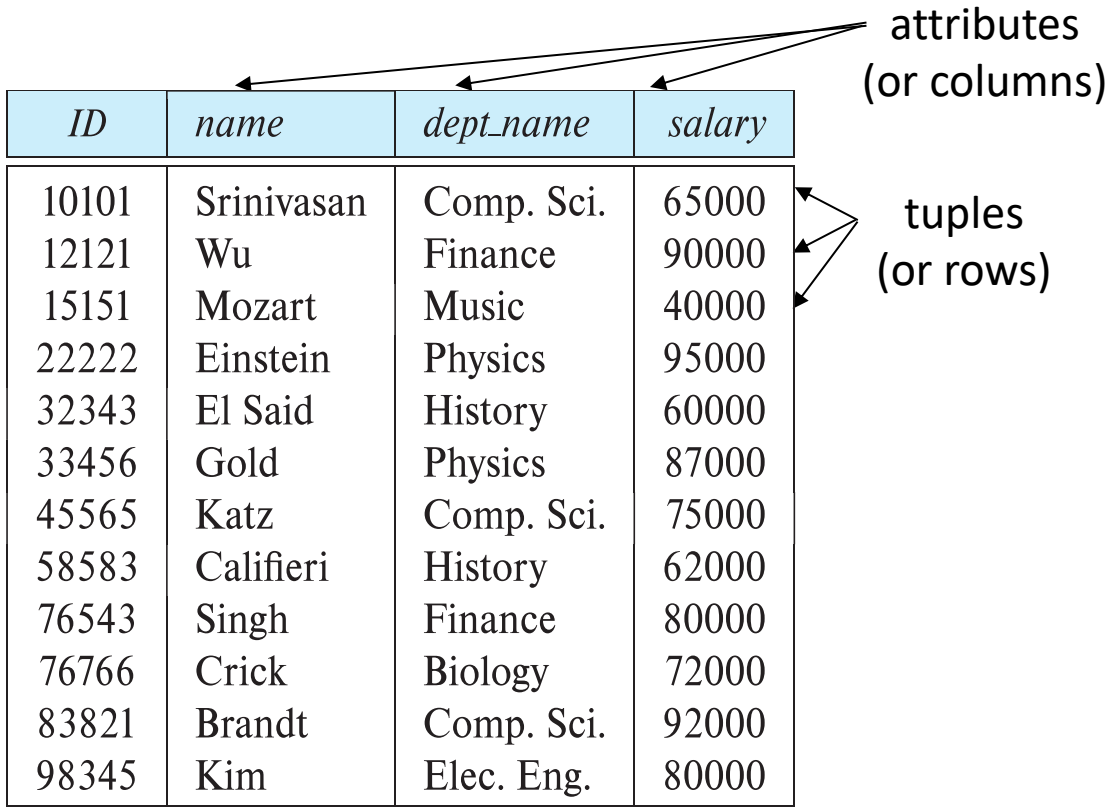


Ted Codd
Turing Award 1981

Relational Schemas

Northeastern University

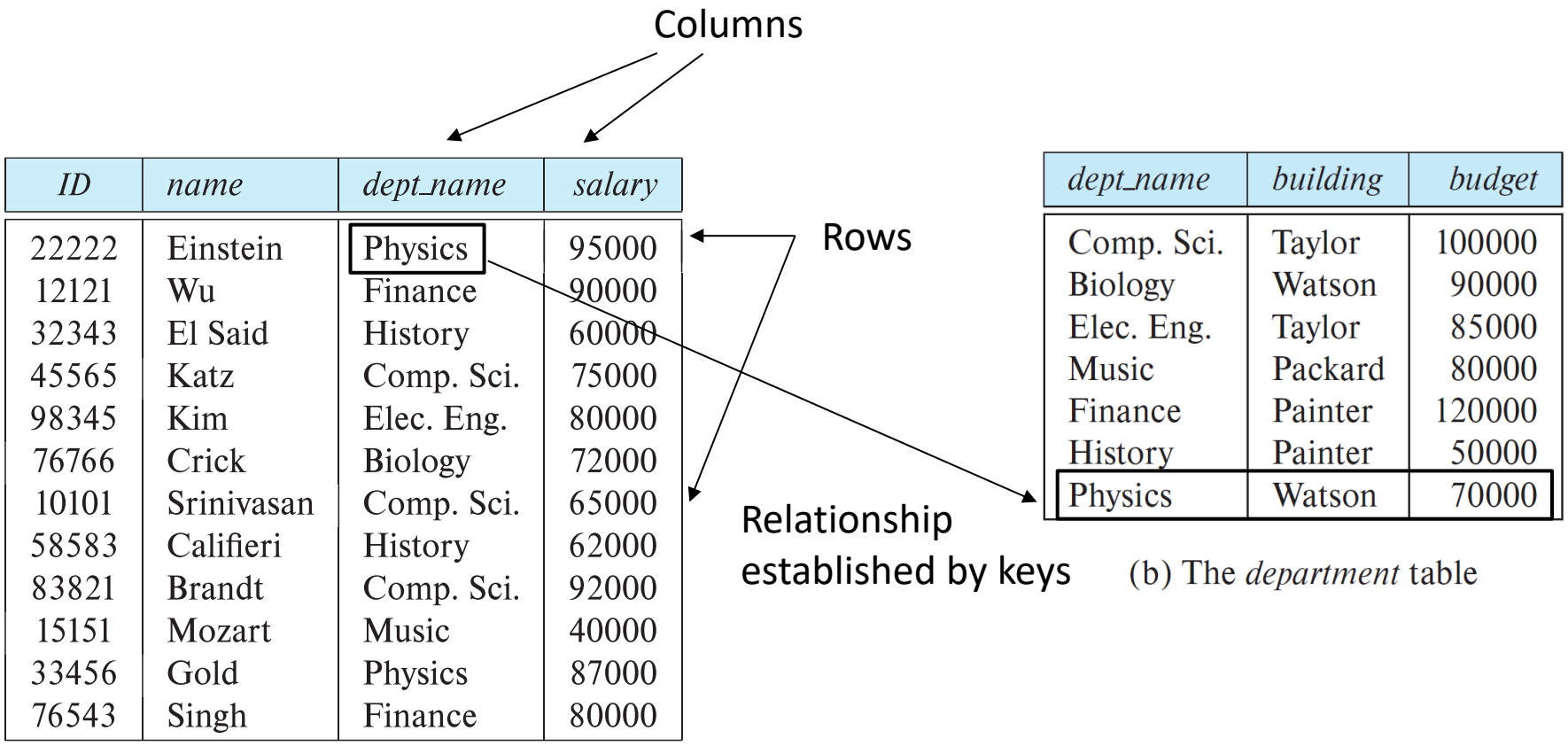
- Example for a relational schema: *instructor* (*ID*, *name*, *dept_name*, *salary*)
  - We often use the same name (e.g., *instructor*) to refer both to the schema and the instance



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Relational Schemas

- Tuples in different relations are "linked" using keys



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

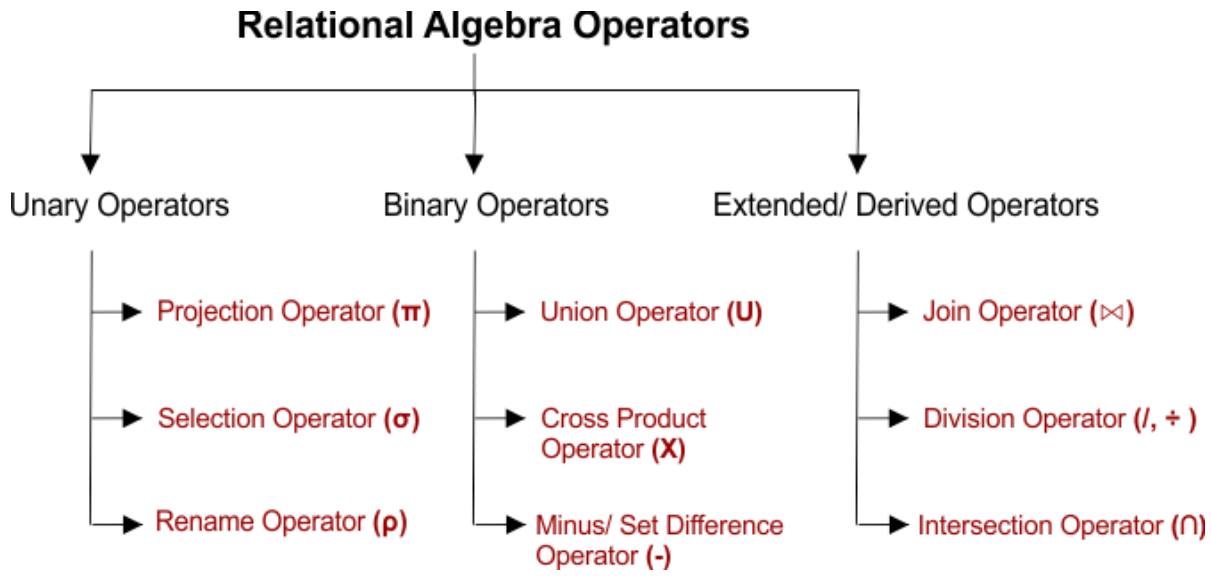
(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Relational Algebra

- Relational algebra is a procedural query language introduced by Edgar F. Codd
- Consists of a set of mathematical operators defined on relations
- Provides a theoretical foundation for relational query languages such as SQL



S (S#, SNAME, CITY, STATUS)

P (P#, PNAME, CITY, WEIGHT, COLOR)

SP (S#, P#, QUANTITY)

Example:

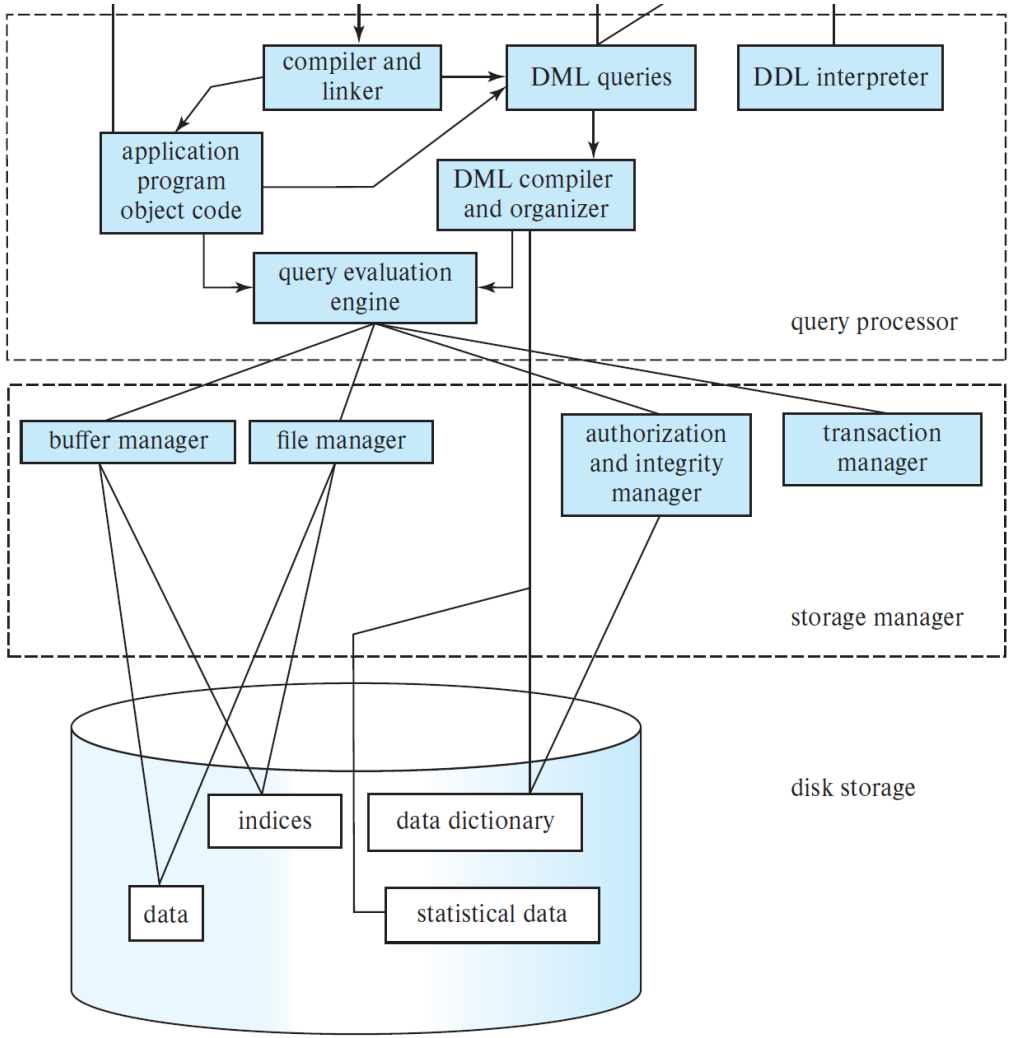
<Supplier names of suppliers who ship "P3" >

$$\Pi_{SNAME}\left(S \bowtie \left(\Pi_{S\#}\left(\sigma_{P\#=3}\, SP\right)\right)\right)$$

DBMS Architecture

Northeastern University

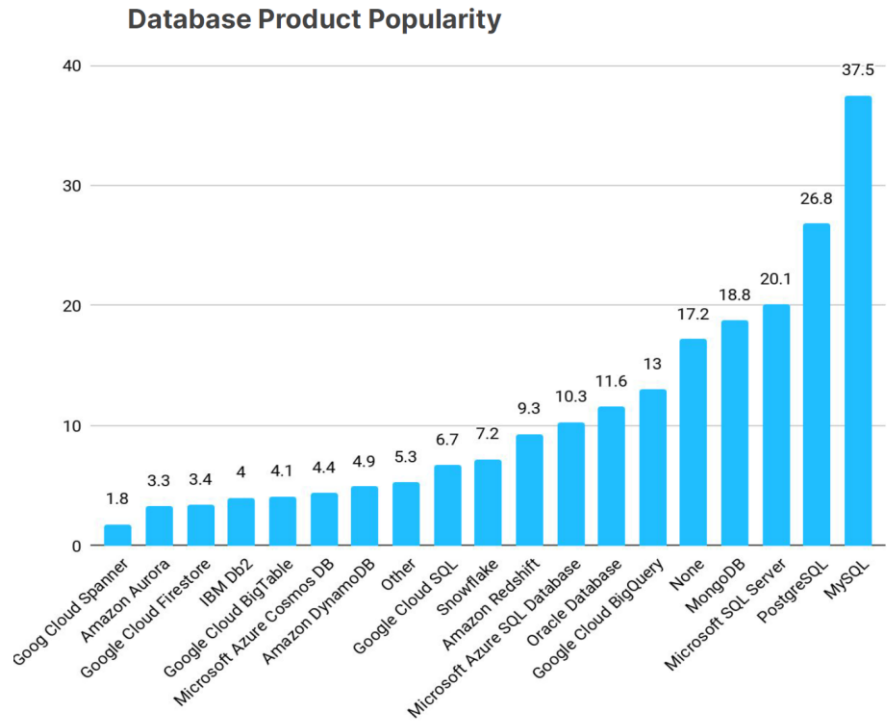
- A DBMS consists of two major components
  - Storage manager
  - Query processor



MySQL

Northeastern University

- MySQL is a highly popular database, offered under two editions
  - Open source MySQL Community Server
  - Enterprise Server (owned by Oracle)

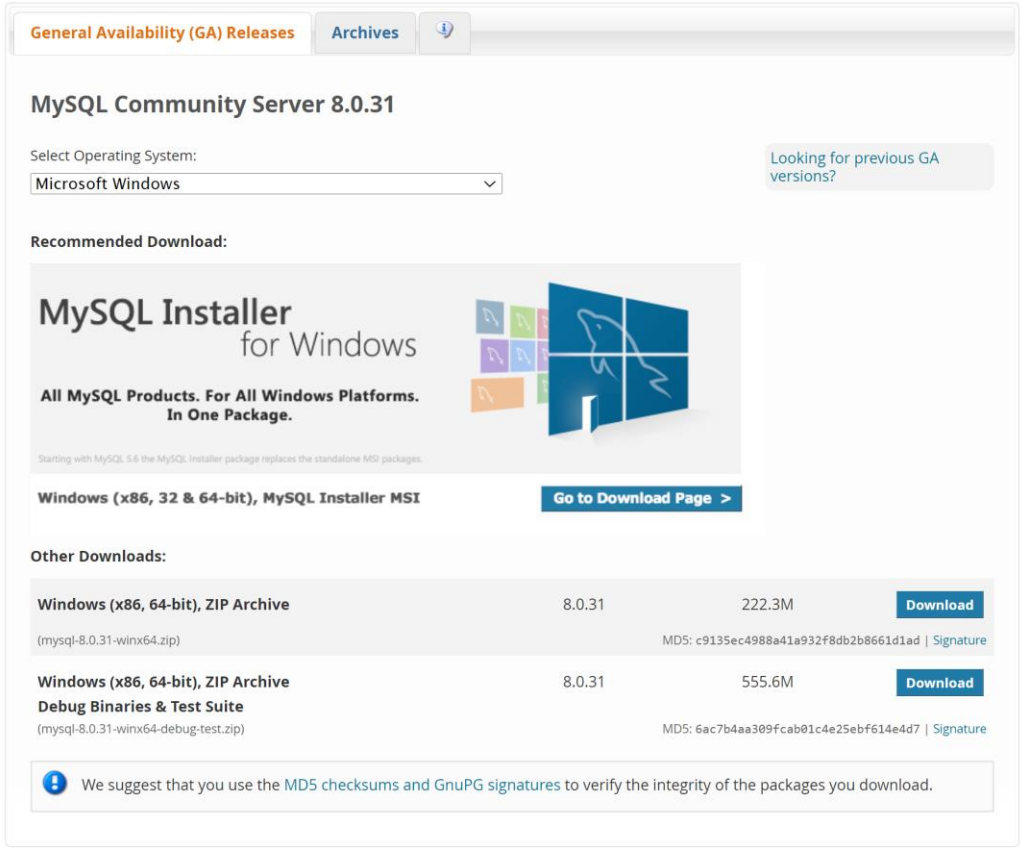


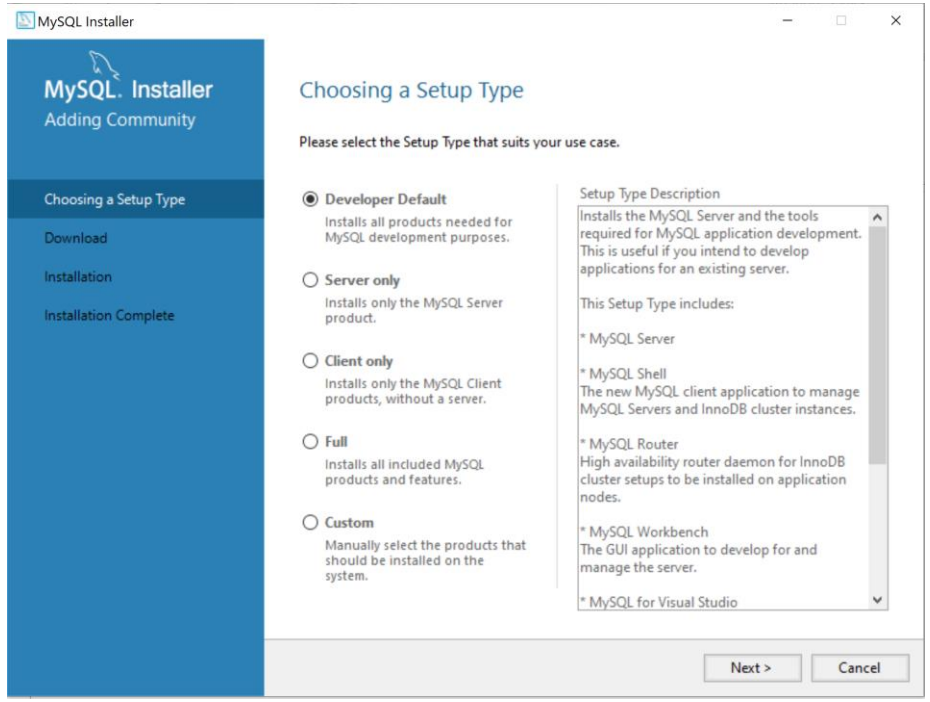
https://www.kaggle.com/kaggle-survey-2021

MySQL Installation

Northeastern University

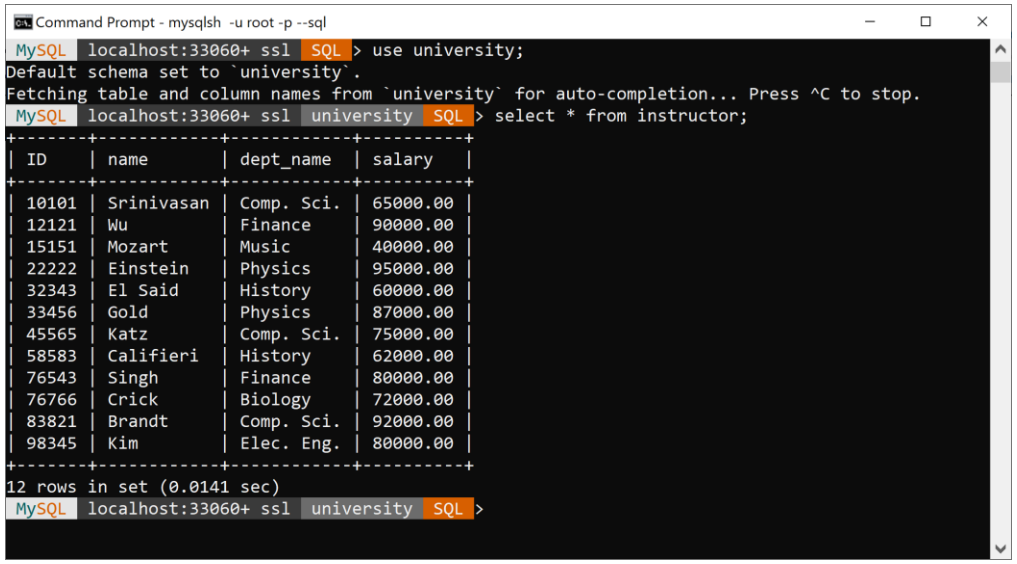
- Download MySQL Community Server from https://dev.mysql.com/downloads/mysql/
- Run the setup with the default settings (remember the password you set for root)

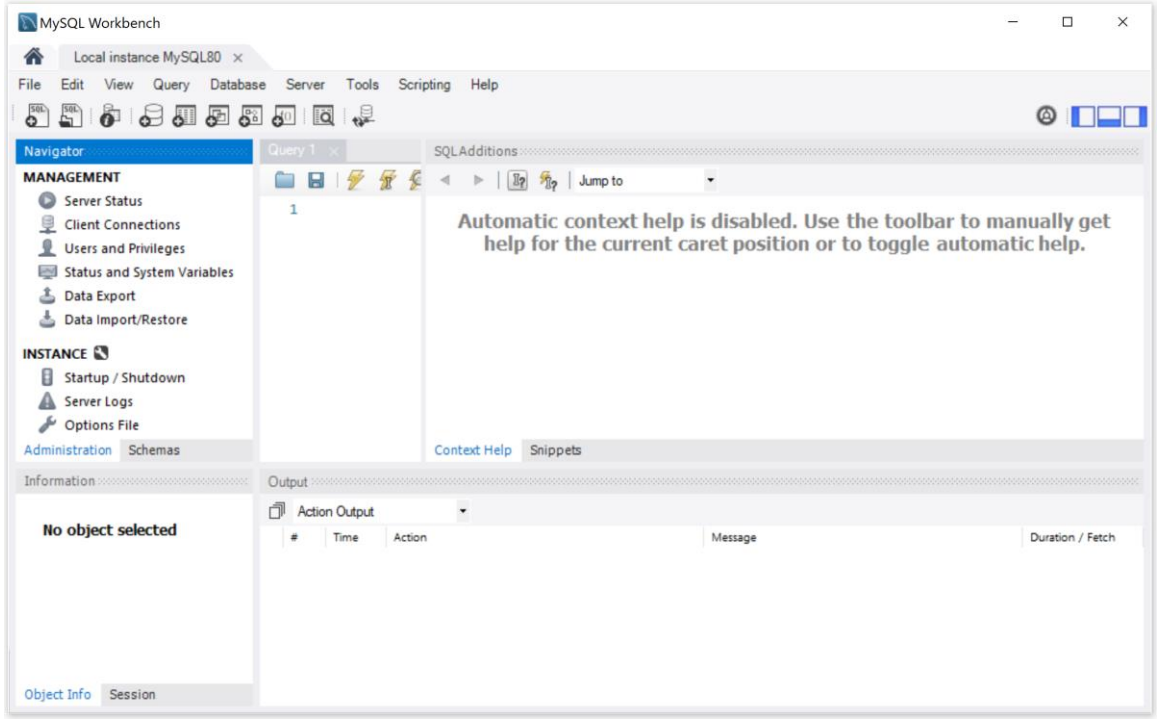




MySQL Tools

- After the installation, you will have two programming tools to work with MySQL:
  - MySQL Shell, where you can type SQL commands directly in the console
  - A graphical user interface tool called **MySQL Workbench**



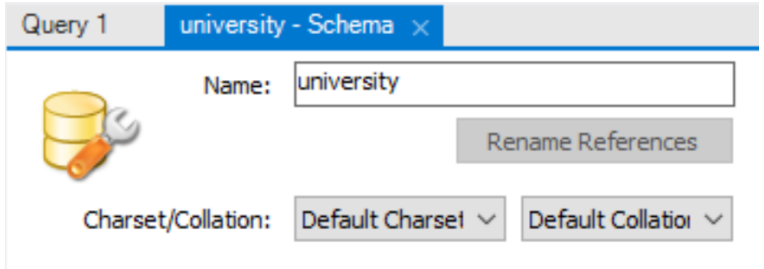


Create a New Database

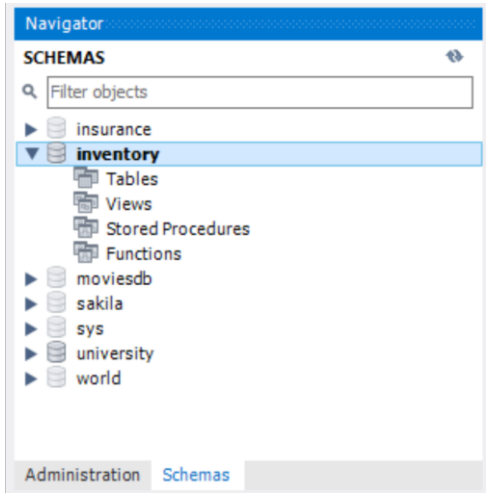
Northeastern University

- Click the Create New Schema button in the toolbar

- Give the database a name

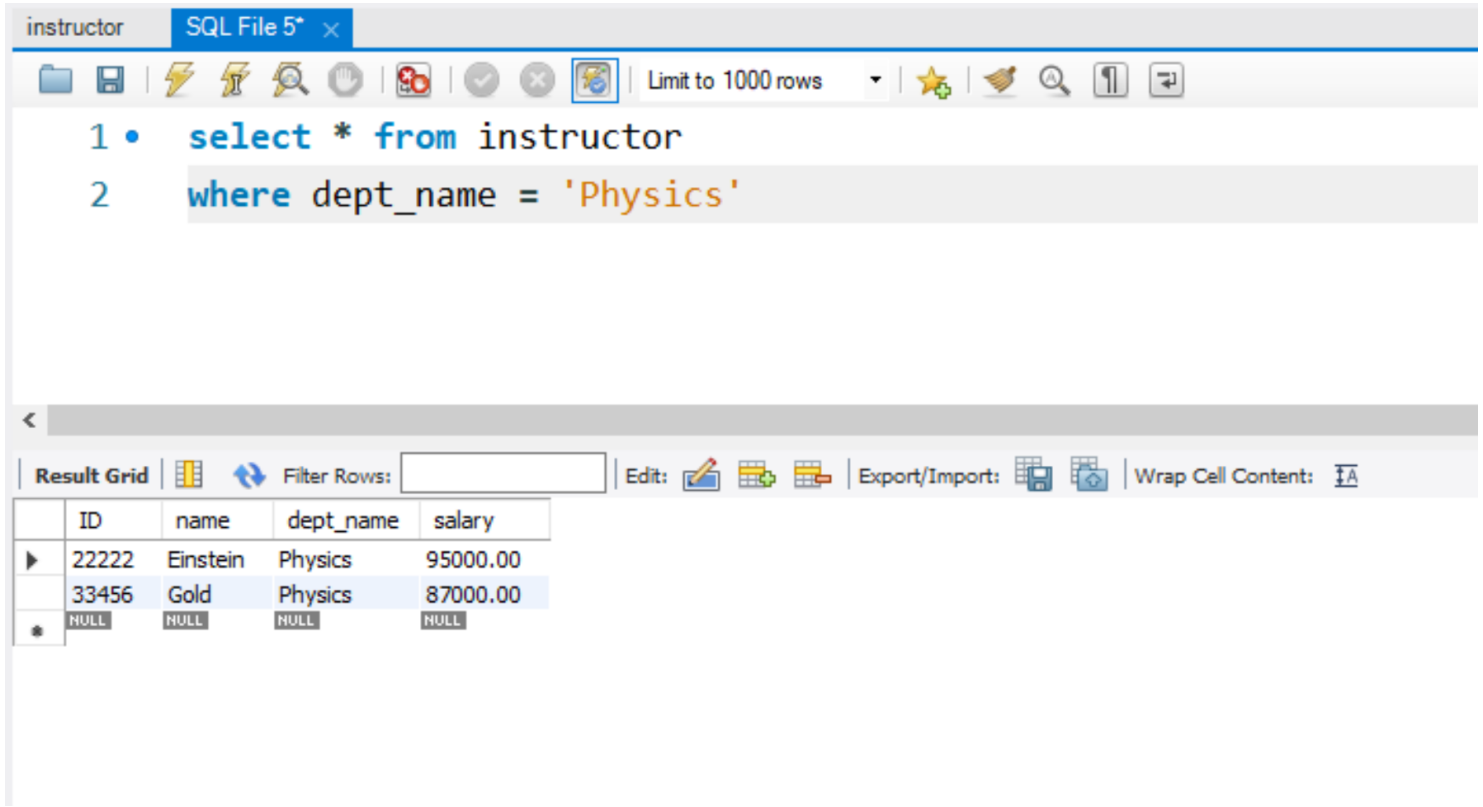


- Double-click on a database to select it
  - All your SQL statements will now be directed to this database
  - Alternatively, you can type USE [database] before your SQL statement



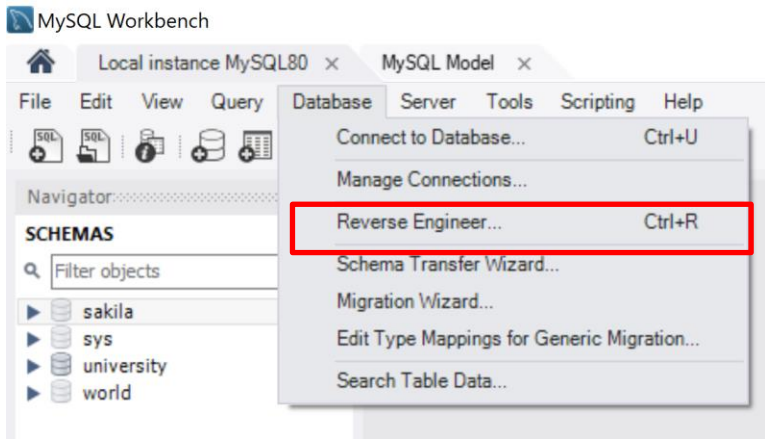
Run SQL Query

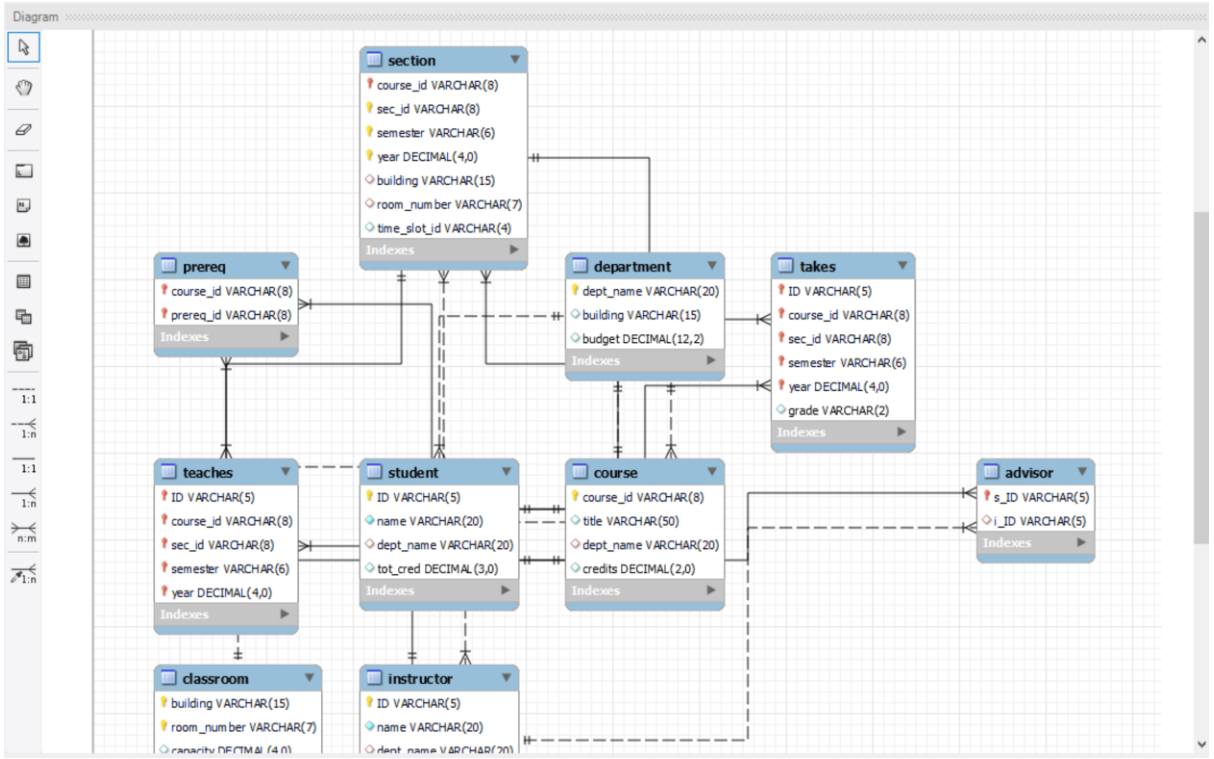
- To run an SQL query, click the Create New SQL button in the toolbar
- Type your query in the editor and then click Execute



Generating Schema Diagrams in MySQL

- A schema diagram depicts the structure of the database graphically





Good Database Design Principles

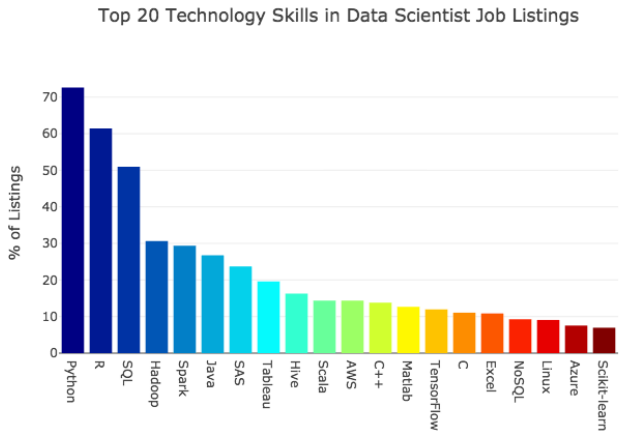
Northeastern University

- Avoid duplicate information (redundancy)
  - e.g., storing the course title in each section of the course
  - Wastes space and can lead to inconsistencies
- Store information in its smallest logical parts
  - e.g., don’t use a single field for a full name
  - Makes it difficult to retrieve individual facts later
- Define a primary key for each table
  - If there is no column that might make a good primary key, use an auto-increment column
- Ensure the integrity of the data by applying integrity constraints
  - e.g., define NOT NULL constraint on columns that cannot be empty
- Apply normalization rules

SQL

Northeastern University

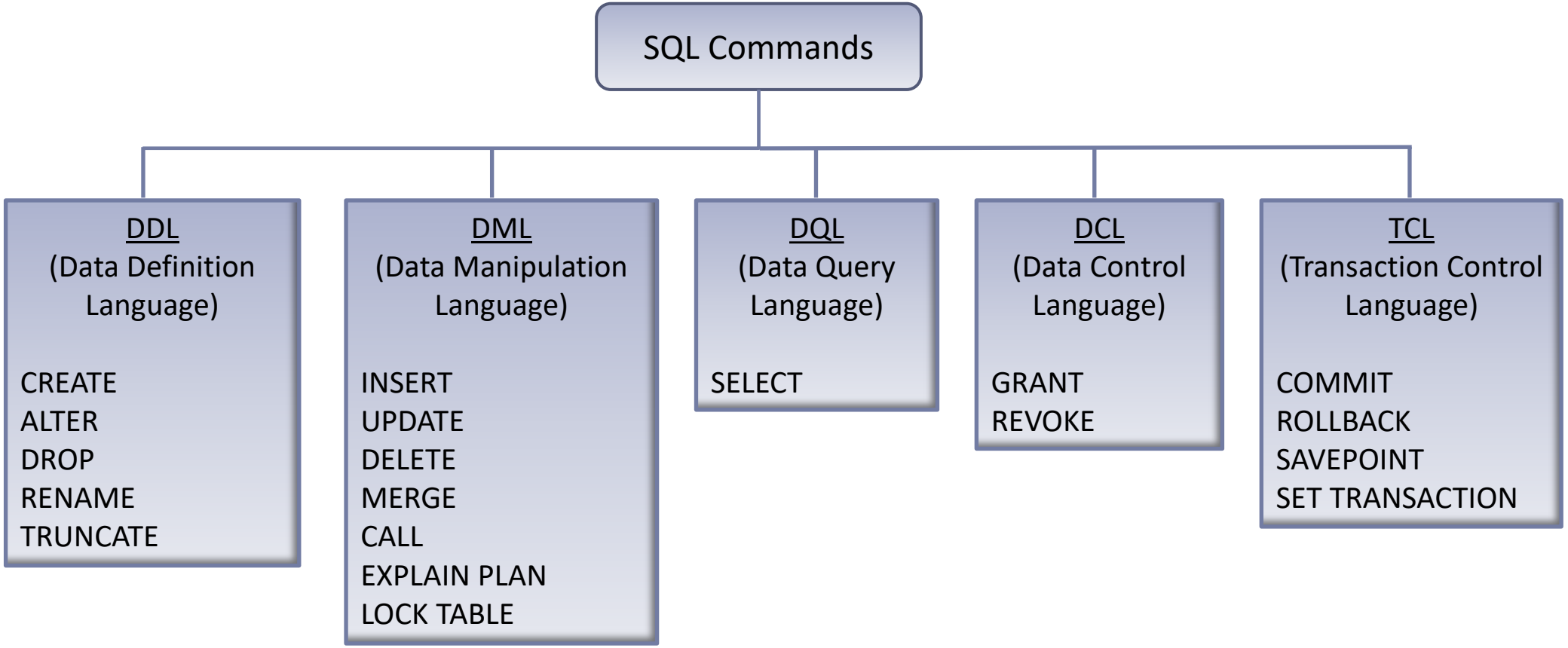
- **Structured Query Language** (SQL) is the standard relational databases language
- SQL is a set-based, declarative programming language
- SQL became an ANSI and ISO standard in 1986, called SQL-86
- Most commercial systems support SQL-92 features
- Many database vendors provide their own extensions to the basic SQL
  - e.g., T-SQL (Microsoft), PL/SQL (Oracle), SQL/PSM (MySQL)



SQL Commands

Northeastern University

- SQL commands can be grouped into the following categories



Basic SQL Syntax Rules

Northeastern University

- SQL is case insensitive
  - e.g., Select ≡ SELECT ≡ select
  - However, in MySQL the case makes difference in table names
- All SQL statements end with a semicolon (;)
  - The semicolon is usually optional when executing a single SQL command
- Strings are specified in **single quotes** only
- In-line comments start with --
- Multi-line comments start with /* and end with */

```
/* This is a multi-line
   comment */
SELECT * FROM customers; -- This is an inline comment
```

- See SQL style guide here

MySQL String Data Types

Northeastern University

Data Type	Description
CHAR(size)	Fixed length string with a user-specified length (can be from 0 to 255)
VARCHAR(maxsize)	Variable-length string with user-specified maximum length (can be from 0 to 65,535)
TINYTEXT	A string with a maximum length of 255 characters
TEXT	A string with a maximum length of 65,535 (~64KB) characters
MEDIUMTEXT	A string with a maximum length of 16,777,215 (~16MB) characters
LONGTEXT	A string with a maximum length of 4,294,967,295 (~4GB) characters

- Differences between VARCHAR and TEXT:
  - You cannot specify the maximum length of TEXT fields
  - VARCHAR is stored inline with the table, while TEXT is usually stored off the table
- In general, you should use VARCHAR for columns below 65,535 characters

MySQL Numeric Data Types

Northeastern University

Data Type	Description
TINYINT(size)	An integer between -128 and 127 The maximum number of digits may be specified in parentheses. BOOLEAN is synonym for TINYINT(1)
SMALLINT(size)	An integer between -32768 and 32767
MEDIUMINT(size)	An integer between -8388608 and 8388607
INT(size)	An integer between -2147483648 and 2147483647
BIGINT(size)	An integer between -9223372036854775808 and 9223372036854775807
FLOAT(size, d)	A single-precision floating point number The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point may be specified in the d parameter.
DOUBLE(size, d)	A double-precision floating point number REAL is synonym for DOUBLE
DECIMAL(size, d)	A double stored as a string, allowing for a fixed decimal point

MySQL Other Data Types

Northeastern University

Data Type	Description
DATE	A date in format: YYYY-MM-DD
DATETIME	A date and time combination. Format: YYYY-MM-DD HH:MI:SS
TIMESTAMP	A timestamp. Timestamp values are stored as the number of seconds since the Unix epoch (1970-01-01 00:00:00 UTC). Format: YYYY-MM-DD HH:MI:SS
TIME	A time. Format: HH:MI:SS
ENUM(x, y, z, …)	Lets you enter a list of possible values. You can list up to 65,535 values in the enum list.
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice.
BLOB	Binary large object. Holds up to 65,535 bytes of data. Can be used to store images.
LONGBLOB	Binary large object. Holds up to 4,294,967,295 bytes of data.

Data Definition Language (DDL)

Northeastern University

- SQL DDL commands allows you define the tables in the database including:
  - The schema of each table
  - The type of values associated with each attribute
  - Integrity constraints (e.g., primary and foreign keys)
  - Indexes
  - The physical storage structure of each table on disk

Creating a Table

Northeastern University

- A table is defined using the **CREATE TABLE** command:

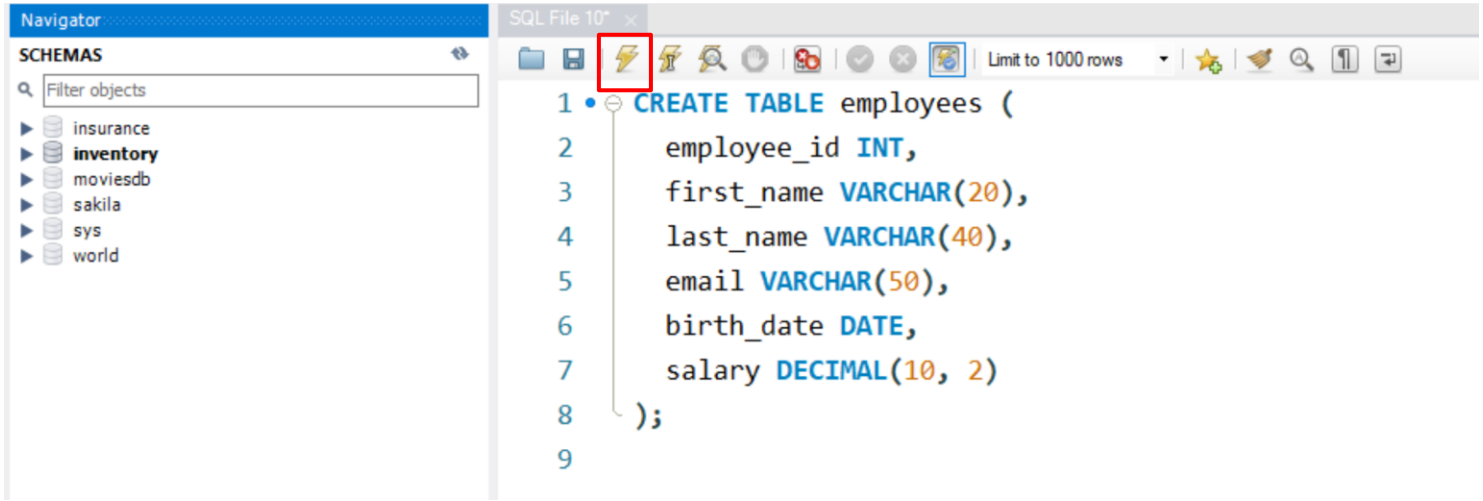
```
CREATE TABLE table_name (
  column1 datatype [ constraints ],
  column2 datatype [ constraints ],
  ...
);
```

- Example:

```
CREATE TABLE employees (
  employee_id INT,
  first_name VARCHAR(20),
  last_name VARCHAR(40),
  email VARCHAR(50),
  birth_date DATE,
  salary DECIMAL(10, 2)
);
```

Creating a Table

- To run SQL commands in Workbench, click the Create New SQL button in the toolbar
- Type your command in the editor
- Make sure you are located in the correct schema (it should be displayed in bold)
  - If it's not the correct schema, double-click on the schema to change it
  - Or type USE [name of database] as the first line of your command
- Then click Execute



Integrity Constraints

Northeastern University

- Ensure that changes to the database don't result in loss of data or consistency
- Declared as part of the CREATE TABLE command
- Types of constraints
  - PRIMARY KEY
  - NOT NULL
  - UNIQUE
  - CHECK
  - FOREIGN KEY

Primary Key

Northeastern University

- The primary key uniquely identifies each record in the table
- A table can have only one primary key
- The primary key can consist of one or more columns
- The columns of the primary key are automatically defined as unique and not null
- The primary key should be chosen such that its values are never/rarely changed
- The primary key can be defined as part of a column declaration or appear on its own

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(20),
  last_name VARCHAR(40),
  email VARCHAR(50),
  birth_date DATE,
  salary DECIMAL(10, 2)
);
```

```
CREATE TABLE employees (
  employee_id INT,
  first_name VARCHAR(20),
  last_name VARCHAR(40),
  email VARCHAR(50),
  birth_date DATE,
  salary DECIMAL(10, 2),
  PRIMARY KEY (employee_id)
);
```

Auto Increment

Northeastern University

- AUTO_INCREMENT can be used to generate a unique identity for new rows
  - It is often defined as an attribute of the primary key of the table

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(20),
  last_name VARCHAR(40),
  email VARCHAR(50),
  birth_date DATE,
  salary DECIMAL(10, 2)
);
```

- By default, the starting value of the field is 1, and increments by 1 for each new row
  - You can change the seed value by writing AUTO_INCREMENT=x at the end of CREATE TABLE

NOT NULL Constraint

Northeastern University

- By default, every column can hold NULL values
- The NOT NULL constraint specifies that a column cannot have NULL values
- This means you cannot insert a new row without specifying a value for this column
- For example ensures that the first or last name of an employee are not NULL:

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(20) NOT NULL,
  last_name VARCHAR(40) NOT NULL,
  email VARCHAR(50),
  birth_date DATE,
  salary DECIMAL(10, 2)
);
```

UNIQUE Constraint

Northeastern University

- Ensures that all the values in a column (or a combination of columns) are unique
- A primary key constraint automatically implies a unique constraint
  - However, you may have many unique constraints per table
- Columns declared as unique are permitted to be NULL
  - Unless they have explicitly been declared to be NOT NULL
- For example, we can declare the email address of an employee to be unique

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(20) NOT NULL,
  last_name VARCHAR(40) NOT NULL,
  email VARCHAR(50) UNIQUE,
  birth_date DATE,
  salary DECIMAL(10, 2)
);
```

CHECK Constraint

Northeastern University

- Used to limit the values that can be placed in a column or a set of column
- May appear as part of the declaration of a column or on its own
- For example, to ensure that the salary of an employee is not negative:

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY AUTO_INCREMENT,
  ...
  salary DECIMAL(10, 2) CHECK (salary >= 0)
);
```

- A more complex check constraint that involves two columns:

```
CREATE TABLE orders (
  order_id INT PRIMARY KEY AUTO_INCREMENT,
  customer_id INT NOT NULL,
  ship_address VARCHAR(60),
  order_date DATETIME,
  shipped_date DATETIME,
  CHECK (shipped_date >= order_date)
);
```

Default Values

Northeastern University

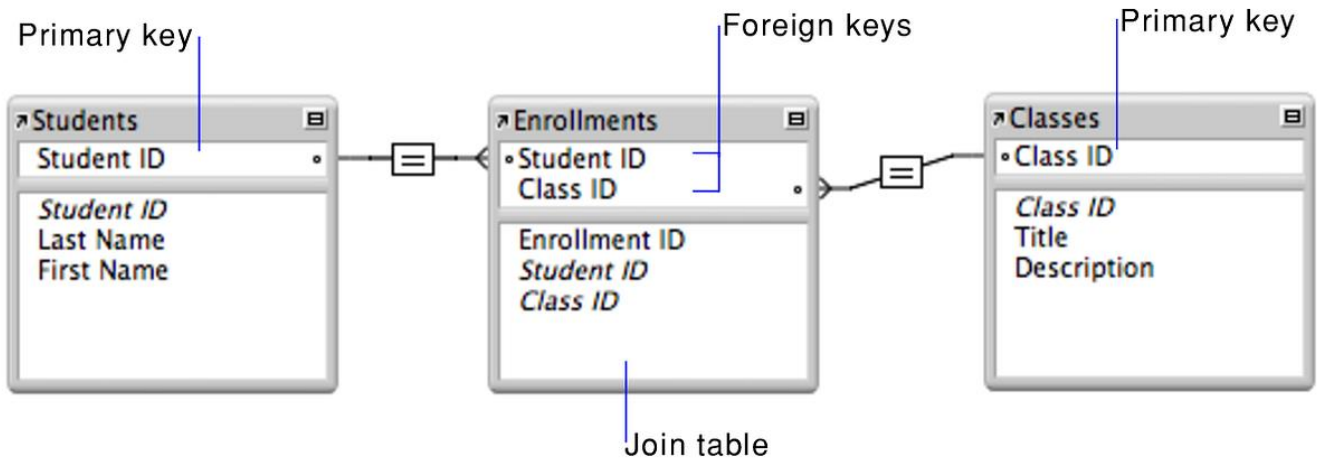
- The DEFAULT constraint can be used to set a default value for a column
- The default value will be added to all new records, if no other value is specified
- The default value can use system functions like now() that returns current time
- For example, we can set the default value of order_date to be now:

```
CREATE TABLE orders (
  order_id INT PRIMARY KEY AUTO_INCREMENT,
  customer_id INT NOT NULL,
  ship_address VARCHAR(60),
  order_date DATETIME DEFAULT NOW(),
  shipped_date DATETIME,
  CHECK (shipped_date >= order_date)
);
```

Foreign Keys

Northeastern University

- A **foreign key** is a column (or set of columns) in one table that refers to a primary key in another table
  - The table with the foreign key is called the **child table**
  - The table with the primary key is called the **parent table**
- Foreign keys enforce **referential integrity** of the data
  - A value in the foreign key column must be one of the values contained in the primary key



Foreign Keys

Northeastern University

- A foreign key constraint is defined using the following syntax:

```
FOREIGN KEY (column1, column2, ...) REFERENCES table(column1, column2, ...)
```

- Example:

```
CREATE TABLE departments (
  department_id INT PRIMARY KEY,
  department_name VARCHAR(20),
  building VARCHAR(15)
);
```

Parent table

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(20) NOT NULL,
  last_name VARCHAR(40) NOT NULL,
  email VARCHAR(50) UNIQUE,
  birth_date DATE,
  salary DECIMAL(10, 2) CHECK (salary >= 0),
  department_id INT,
  FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

Child table

Cascading Actions in Foreign Keys

Northeastern University

- By default, when a referential integrity constraint is violated, the action is rejected
  - e.g., you cannot delete or update a row in the parent table if it has child rows linked to it
- However, the foreign key constraint allows you to define CASCADE rules
  - ON DELETE CASCADE – when a row in the parent table is deleted, automatically delete its corresponding child rows
  - ON UPDATE CASCADE – when the primary key column in the parent table is updated, automatically update the foreign key column in the corresponding child rows

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY AUTO_INCREMENT,
  ...
  department_id INT,
  FOREIGN KEY (department_id) REFERENCES departments(department_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Modifying a Table Definition

Northeastern University

- ALTER TABLE allows you change the table definition
- To add a column to a table, use the following:

```
ALTER TABLE table_name
ADD column_name datatype;
```

- To delete a column from a table, use the following:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

- To change the data type of a column, use the following:

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Dropping Tables

Northeastern University

- The DROP TABLE statement allows you to drop an existing table

```
DROP TABLE table_name;
```

  - Deletes both the data and the schema of the table
- TRUNCATE TABLE deletes only the data in the table, but not the table itself

```
TRUNCATE TABLE table_name;
```

  - It's faster than DELETE if you want to delete all the records from the table

DML Commands

Northeastern University

- SQL DML commands allows you to manipulate the data in the database
  - Insertion of new rows into a given table
  - Deletion of rows in a given table
  - Updating values in a given table

INSERT INTO

Northeastern University

- The INSERT INTO statement is used to insert new records into a table
- There are two ways to write the INSERT INTO statement
  - Specify both the column names and the values to be inserted to these columns

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

  - Provide values for all the columns in the table, without specifying the column names

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

    - The order of the values must be in the same order as the columns in the table
    - You can use NULL to skip a value for a specific column
    - You can use DEFAULT for an auto-increment field or to use the column's default value (if defined)

INSERT INTO Examples

Northeastern University

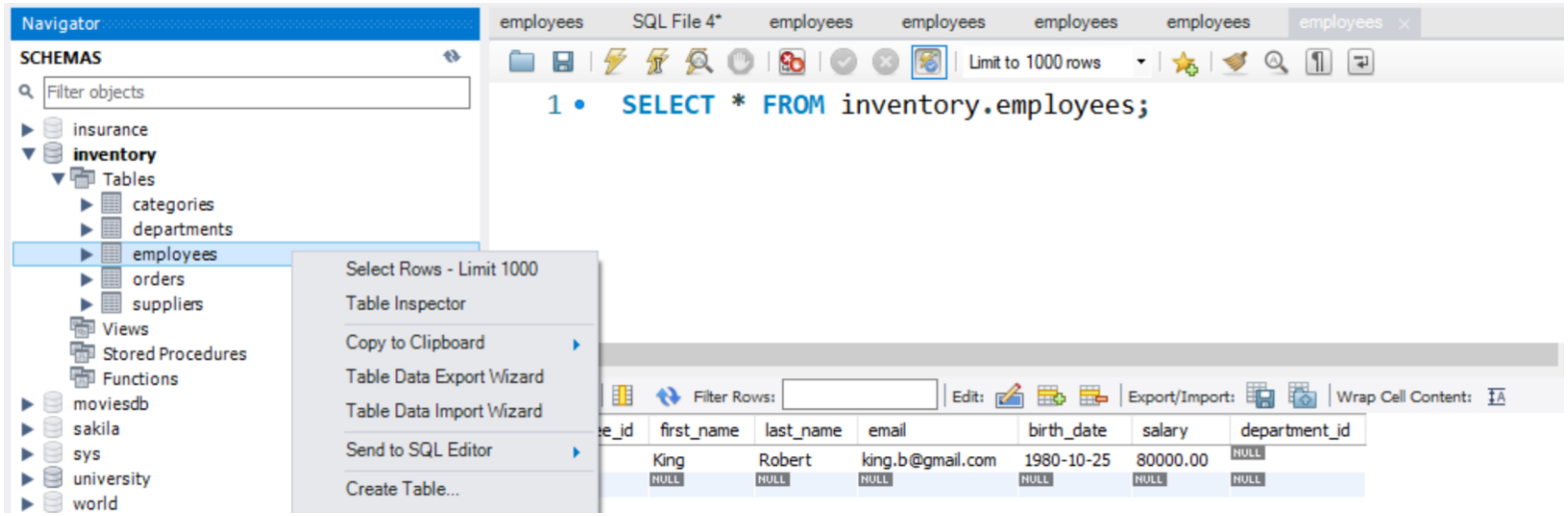
- Adding a new employee to the employees table:

```
INSERT INTO employees
VALUES (DEFAULT, 'King', 'Robert', 'king.b@gmail.com', '1980-10-25', 80000, NULL);
```

- or equivalently

```
INSERT INTO employees (first_name, last_name, email, birth_date, salary)
VALUES ('King', 'Robert', 'king.b@gmail.com', '1980-10-25', 80000);
```

- You can view the data in the table by right clicking on it and select Rows – Limit 1000



UPDATE

Northeastern University

- The UPDATE statement is used to modify existing records in a table

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- The WHERE clause specifies which record(s) should be updated
  - If you omit the WHERE clause, all the records in the table will be updated!
- For example, let's update the salary of employee no. 1 to 90,000:

```
UPDATE employees
SET salary = 90000
WHERE employee_id = 1;
```

- And then give him 10% raise:

```
UPDATE employees
SET salary = salary * 1.1
WHERE employee_id = 1;
```

	employee_id	first_name	last_name	email	birth_date	salary	department_id
▶	1	King	Robert	king.b@gmail.com	1980-10-25	99000.00	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

DELETE

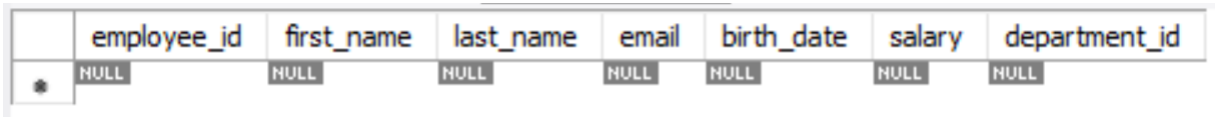
Northeastern University

- The DELETE statement is used to delete records from a table

```
DELETE FROM table_name
WHERE condition;
```

- The WHERE clause specifies which record(s) should be deleted
  - If you omit the WHERE clause, all the records in the table will be deleted!
- Example: delete employee no. 1

```
DELETE FROM employees
WHERE employee_id = 1;
```



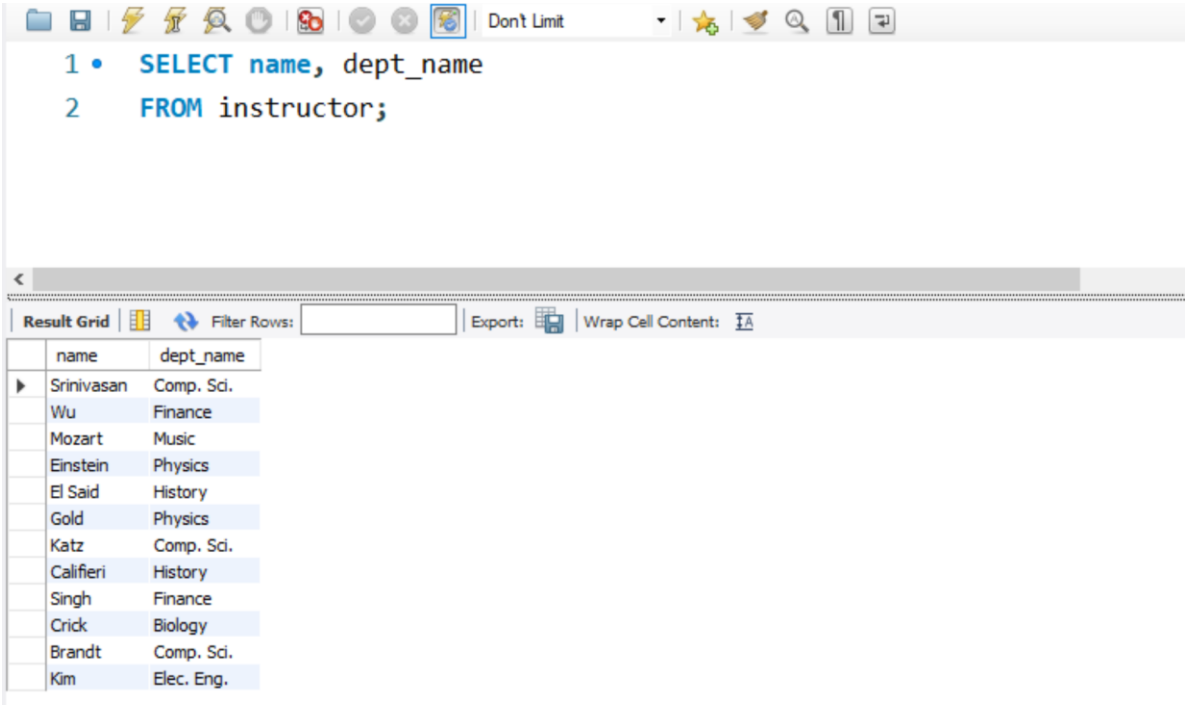
	employee_id	first_name	last_name	email	birth_date	salary	department_id
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SELECT Statement

Northeastern University

- The SELECT statement is used to select data from a table (or a set of tables)
- Basic syntax of SELECT is:

```
SELECT column1, column2, ...
FROM table_name;
```



name	dept_name
Srinivasan	Comp. Sci.
Wu	Finance
Mozart	Music
Einstein	Physics
El Said	History
Gold	Physics
Katz	Comp. Sci.
Califieri	History
Singh	Finance
Crick	Biology
Brandt	Comp. Sci.
Kim	Elec. Eng.

SELECT Statement

Northeastern University

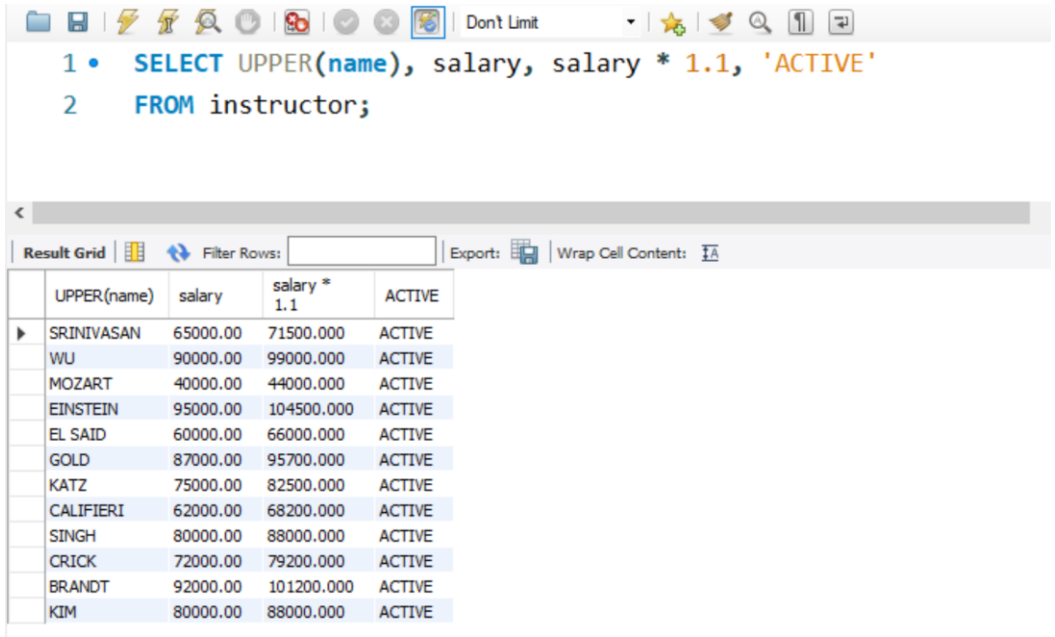
- To select all the columns in the table, use SELECT *

- In general, you should avoid using SELECT *
  - Produces unnecessary I/O and network traffic between the database and the application
  - May expose sensitive information to unauthorized users

The SELECT Clause

Northeastern University

- In addition to table columns, you can include in the SELECT clause other things
  - Literals, such as numbers or strings
  - Arithmetic expression, such as unit_price * 10
  - Built-in function calls
    - For a full list of built-in function see https://dev.mysql.com/doc/refman/8.0/en/functions.html



```
1 •  SELECT UPPER(name), salary, salary * 1.1, 'ACTIVE'
2    FROM instructor;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

UPPER(name)	salary	salary * 1.1	ACTIVE
SRINIVASAN	65000.00	71500.000	ACTIVE
WU	90000.00	99000.000	ACTIVE
MOZART	40000.00	44000.000	ACTIVE
EINSTEIN	95000.00	104500.000	ACTIVE
EL SAID	60000.00	66000.000	ACTIVE
GOLD	87000.00	95700.000	ACTIVE
KATZ	75000.00	82500.000	ACTIVE
CALIFIERI	62000.00	68200.000	ACTIVE
SINGH	80000.00	88000.000	ACTIVE
CRICK	72000.00	79200.000	ACTIVE
BRANDT	92000.00	101200.000	ACTIVE
KIM	80000.00	88000.000	ACTIVE

SQL Aliases

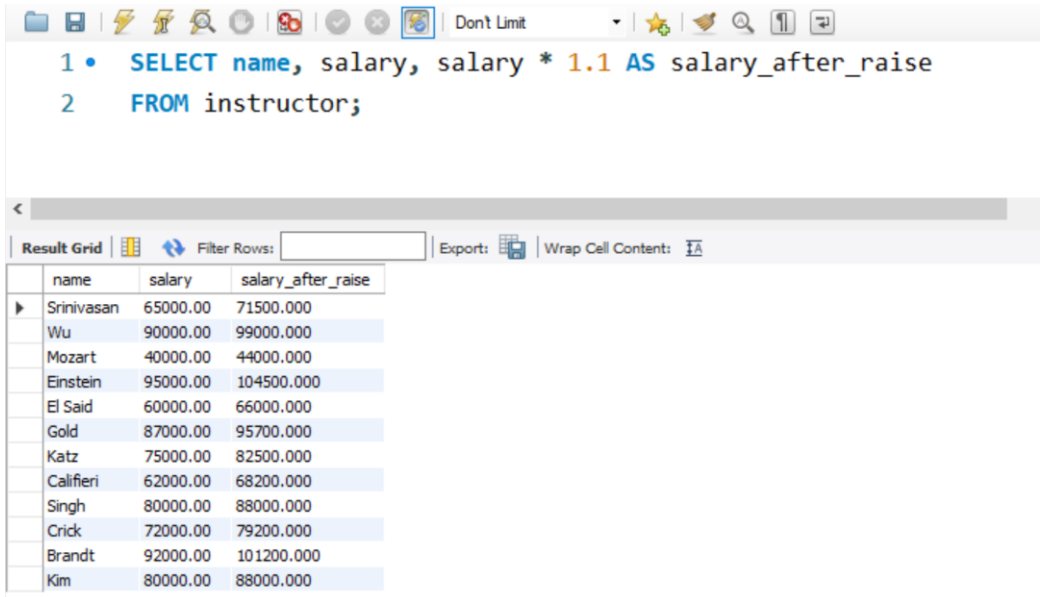
Northeastern University

- SQL aliases are used to rename a table or a column in a specific SQL query
- The syntax for a column alias is:

```
SELECT column_name AS alias_name
FROM table_name;
```

- The syntax for a table alias is:

```
SELECT column_name(s)
FROM table_name AS alias_name;
```



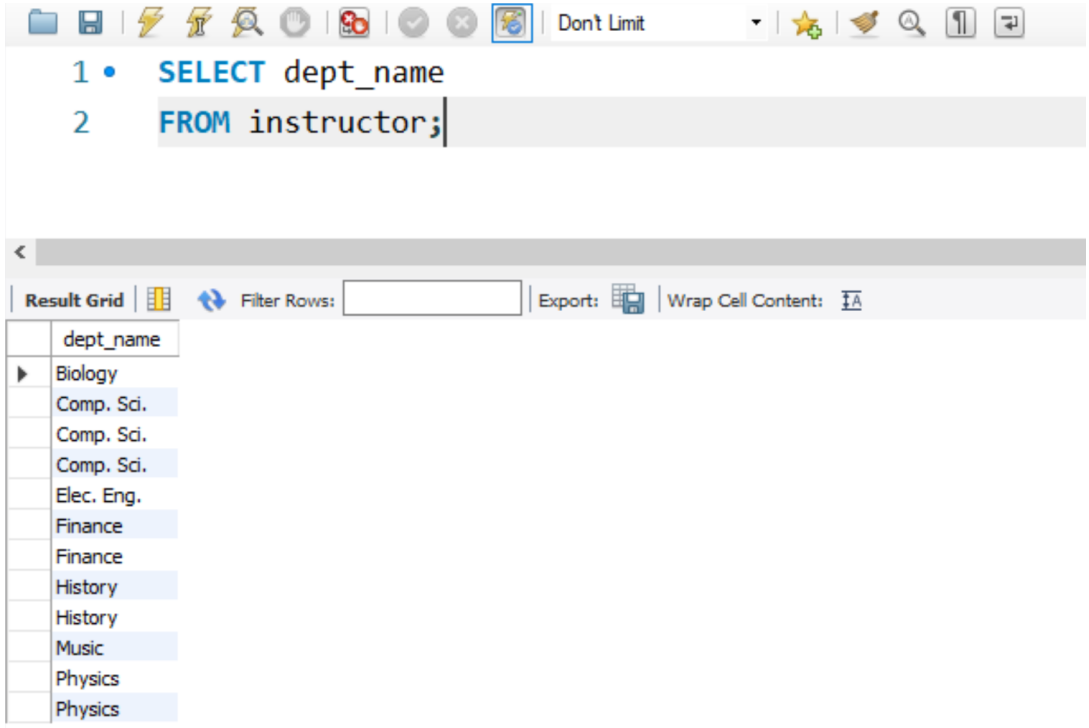
name	salary	salary_after_raise
Srinivasan	65000.00	71500.000
Wu	90000.00	99000.000
Mozart	40000.00	44000.000
Einstein	95000.00	104500.000
El Said	60000.00	66000.000
Gold	87000.00	95700.000
Katz	75000.00	82500.000
Califieri	62000.00	68200.000
Singh	80000.00	88000.000
Crick	72000.00	79200.000
Brandt	92000.00	101200.000
Kim	80000.00	88000.000

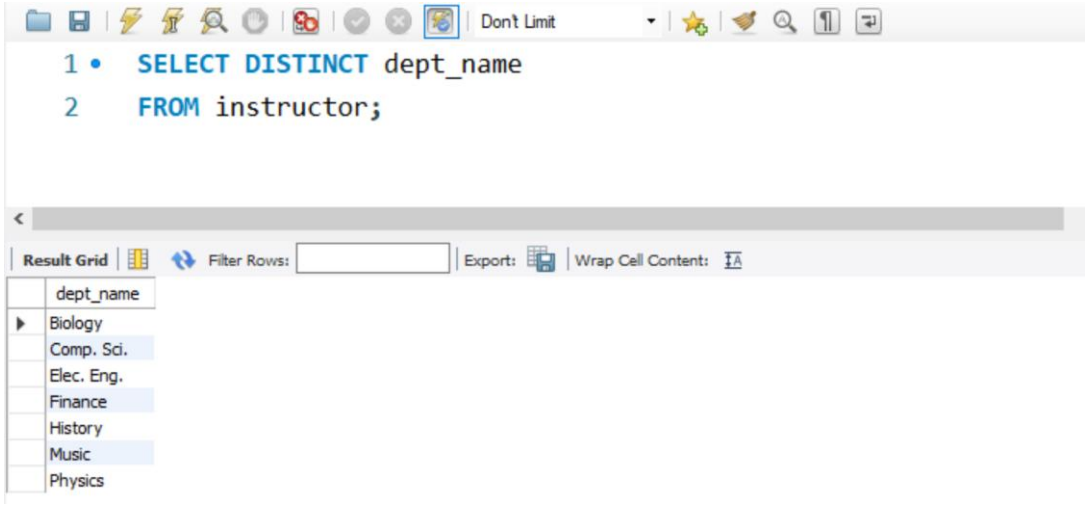
- Aliases are useful when:
  - Functions are used in the query
  - Column names are too long or not very readable
  - Two or more columns are combined together
  - There is more than one table involved in a query

DISTINCT

- In some cases, a query might return duplicate rows of data
- The DISTINCT clause can be used to return only the distinct values from the query

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```





CASE Expression

Northeastern University

- Evaluates a list of conditions and returns a value when the first condition is met

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

```
1 • SELECT id, course_id, grade,
2   CASE
3     WHEN grade IN ('A-', 'A', 'A+') THEN 'Excellent'
4     WHEN grade IN ('B-', 'B', 'B+') THEN 'Good'
5     ELSE 'Bad'
6   END AS status
7   FROM takes;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	course_id	grade	status
00128	CS-101	A	Excellent
00128	CS-347	A-	Excellent
12345	CS-101	C	Bad
12345	CS-190	A	Excellent
12345	CS-315	A	Excellent
12345	CS-347	A	Excellent
19991	HIS-351	B	Good
23121	FIN-201	C+	Bad
44553	PHY-101	B-	Good
45678	CS-101	F	Bad
45678	CS-101	B+	Good

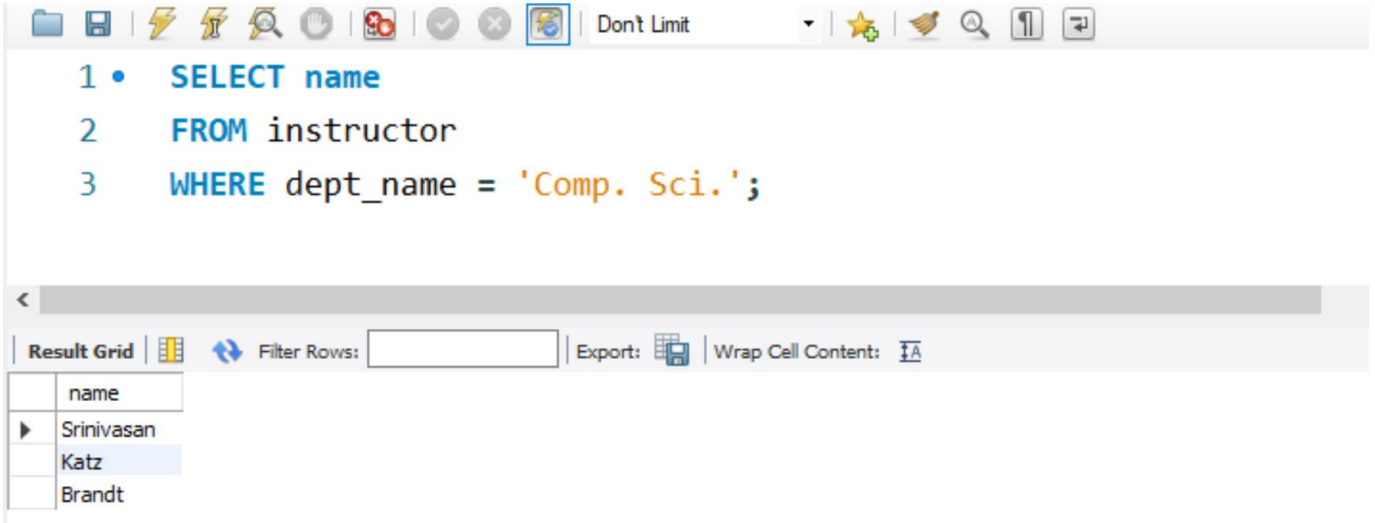
The WHERE Clause

Northeastern University

- The WHERE clause is used to filter records
- It specifies conditions that the result set must satisfy

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Example: Find the names of all instructors in the Comp. Sci. department



The WHERE Clause

Northeastern University

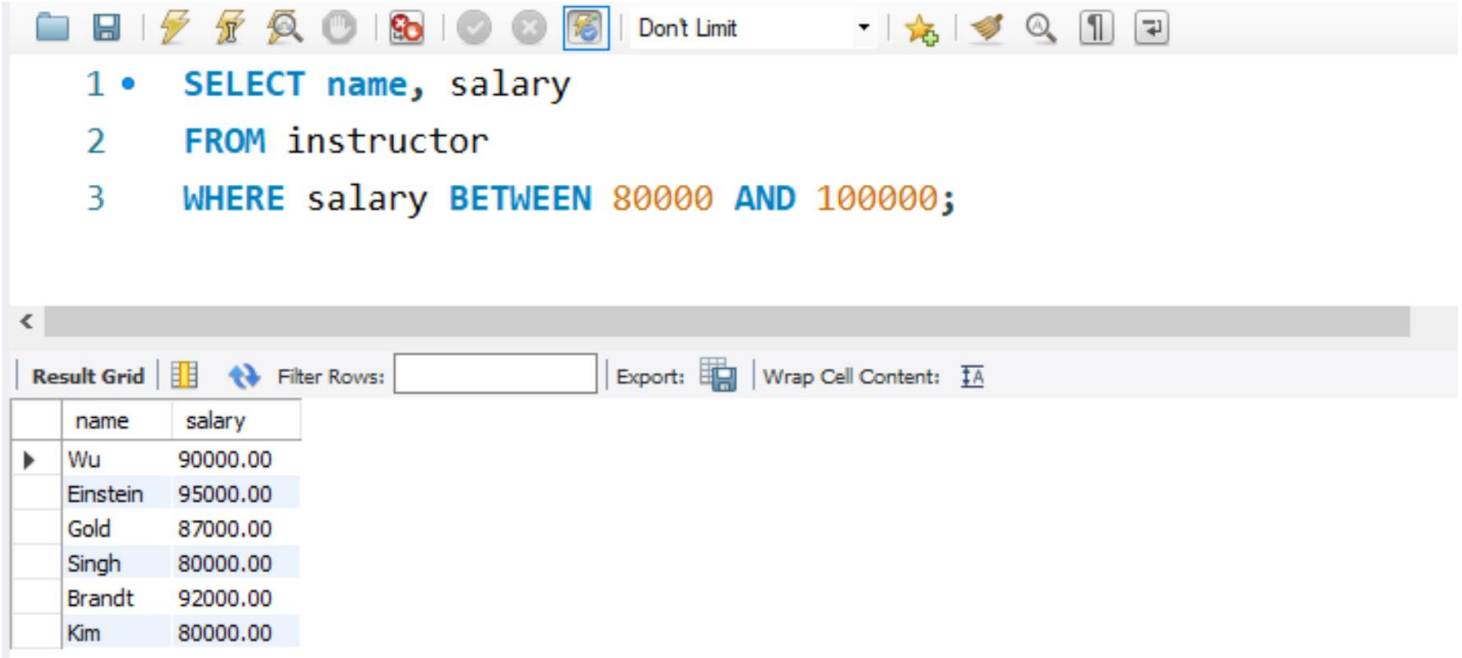
- Operators that can be used in WHERE clause:

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal
BETWEEN	Between a certain range
IN	Test if a value is within a list of values
LIKE	Search for a pattern in a string value

The BETWEEN Operator

Northeastern University

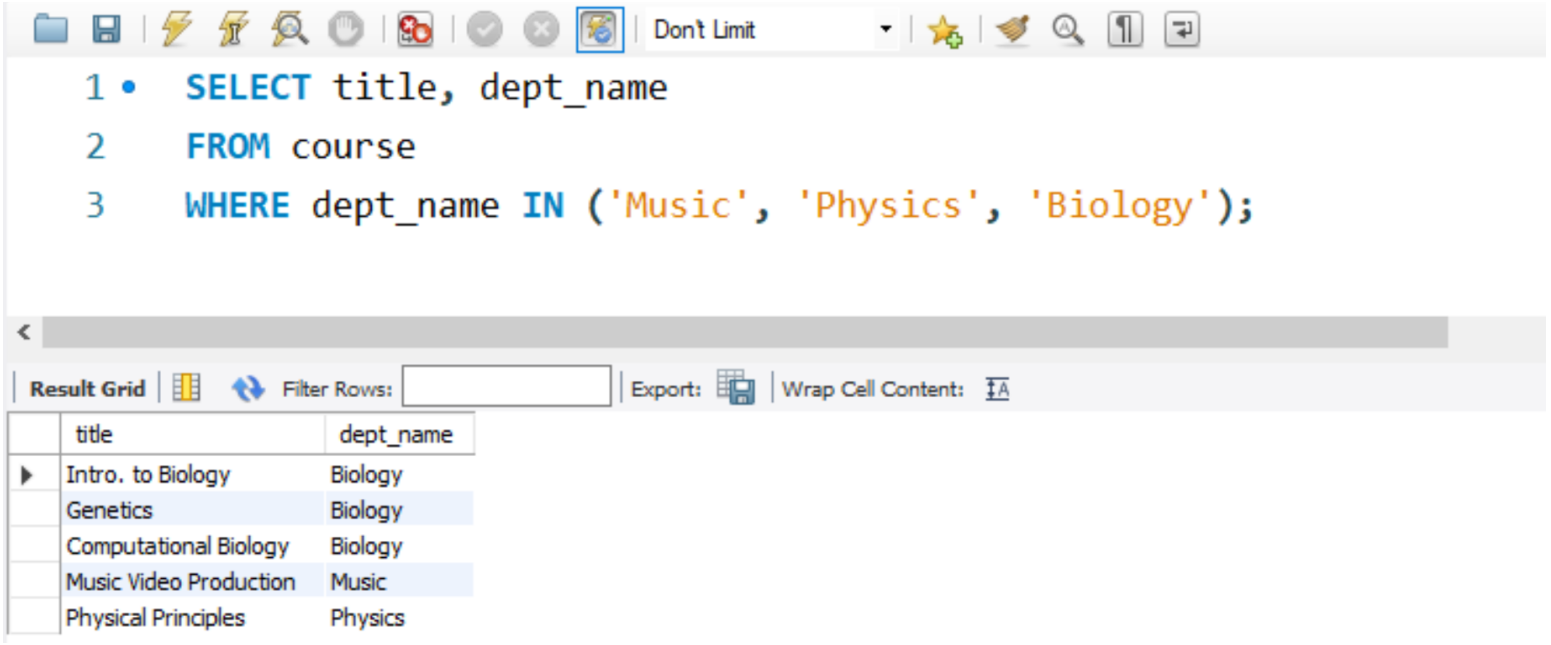
- The BETWEEN operator selects values within a given range
  - The values can be numbers, text, or dates
  - The BETWEEN operator is inclusive: begin and end values are included
- Example: The name and salary of instructors with a salary between 80K and 100K



The IN Operator

Northeastern University

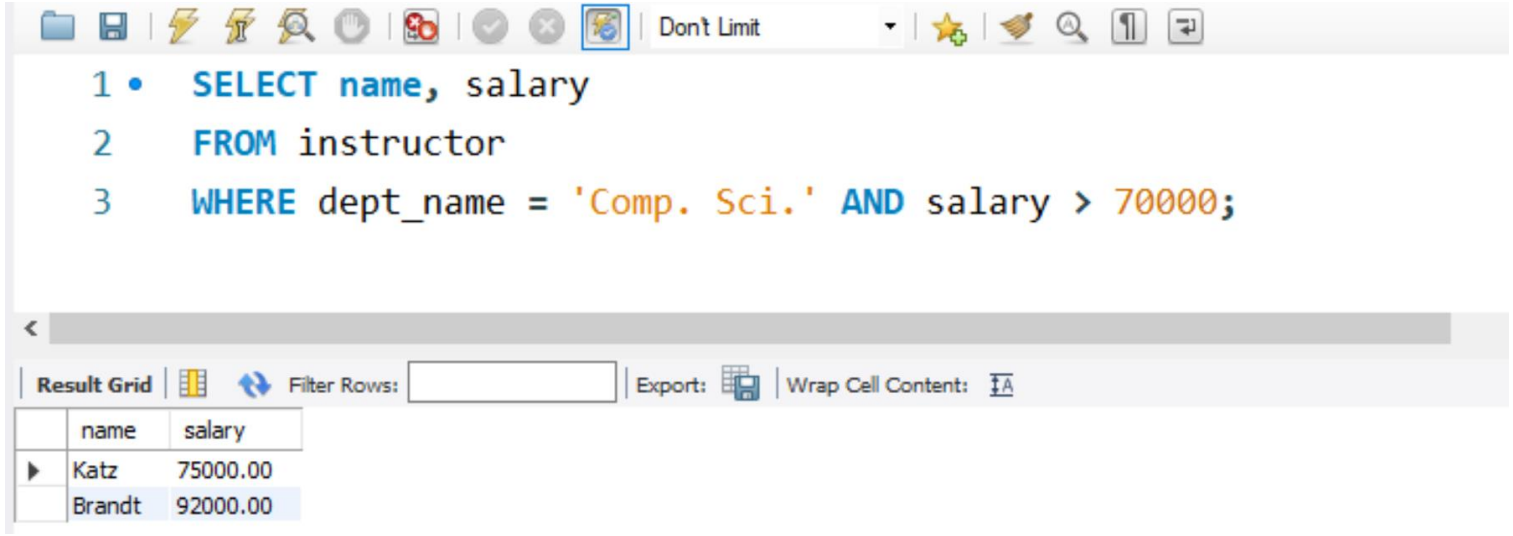
- The IN operator allows you to specify multiple values in the WHERE clause
- Example: Print the title and dept_name of all the courses in the Music, Physics and Biology departments



Logical Operators

Northeastern University

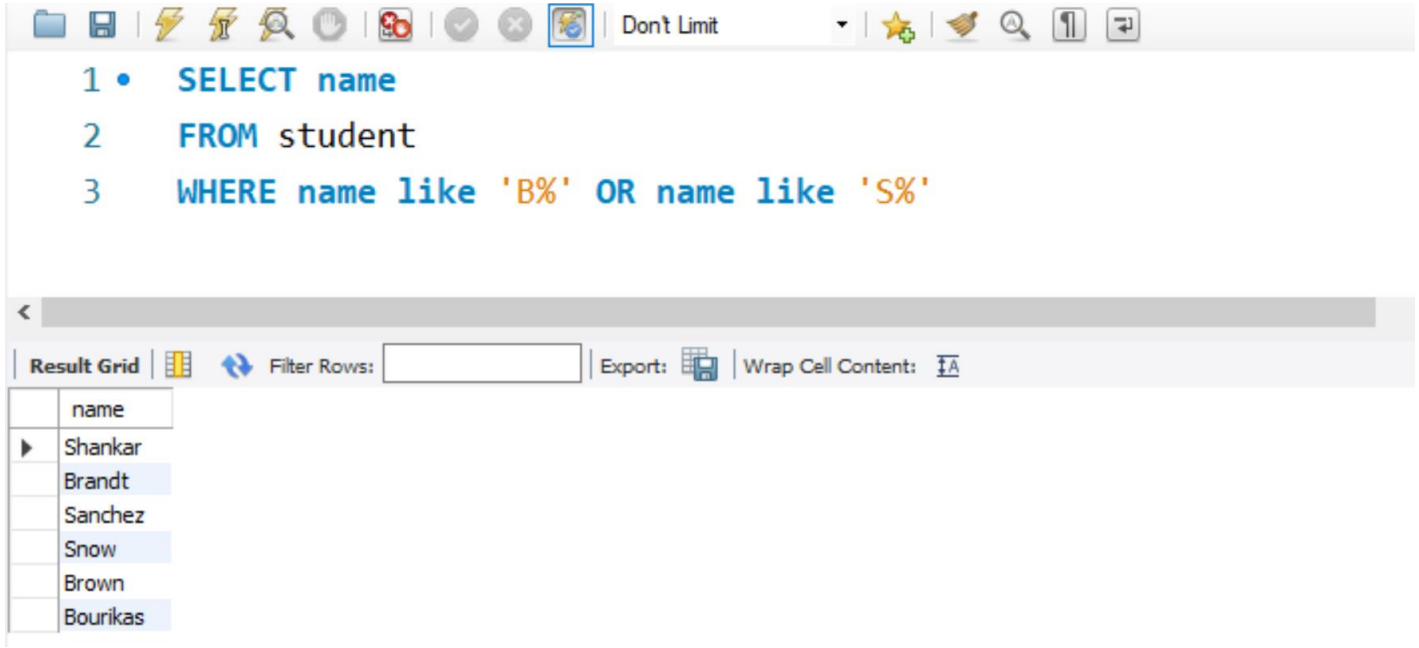
- The WHERE clause can use the logical operators AND, OR and NOT



The LIKE Operator

Northeastern University

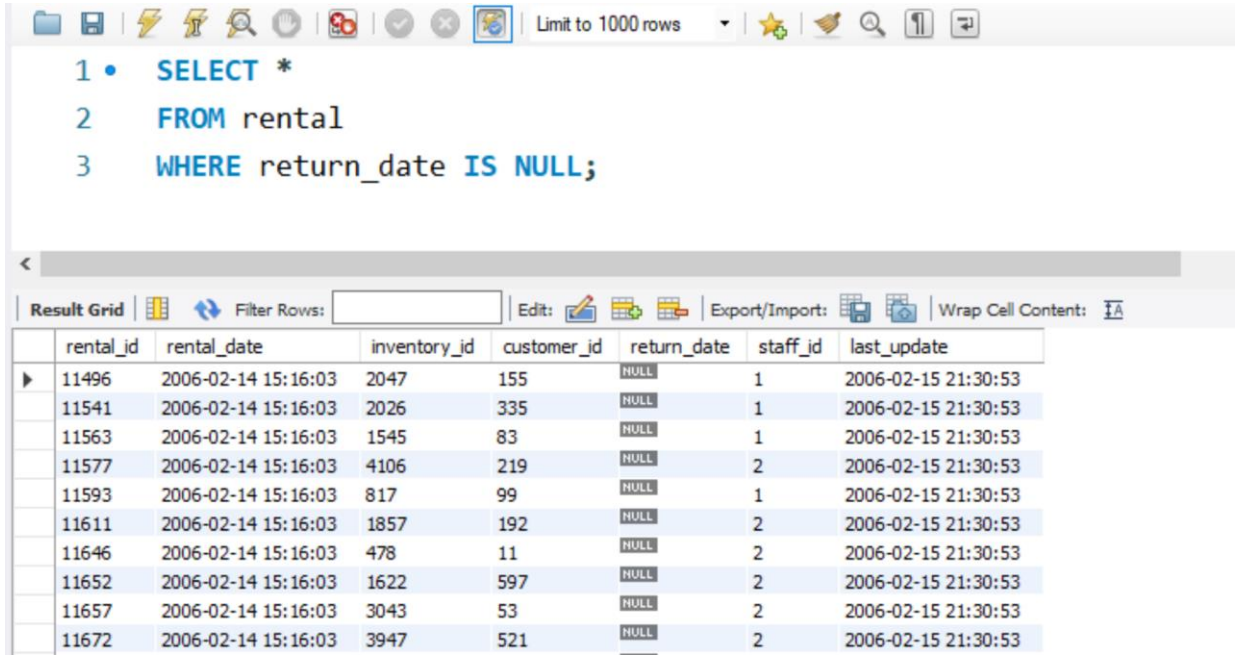
- The LIKE operator allows you to perform pattern matching on strings
- Can use two wildcard characters for constructing patterns:
  - percentage (%) matches any string of 0 or more characters
  - underscore (_) matches any single character



NULL Values

Northeastern University

- A field with a NULL value is a field that has no value
- The result of any arithmetic expression involving NULL is NULL
- Comparing NULL values with operators such as =, < returns an unknown result
- You have to use the **IS NULL** and **IS NOT NULL** operators instead



```
1 • SELECT *
2   FROM rental
3   WHERE return_date IS NULL;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

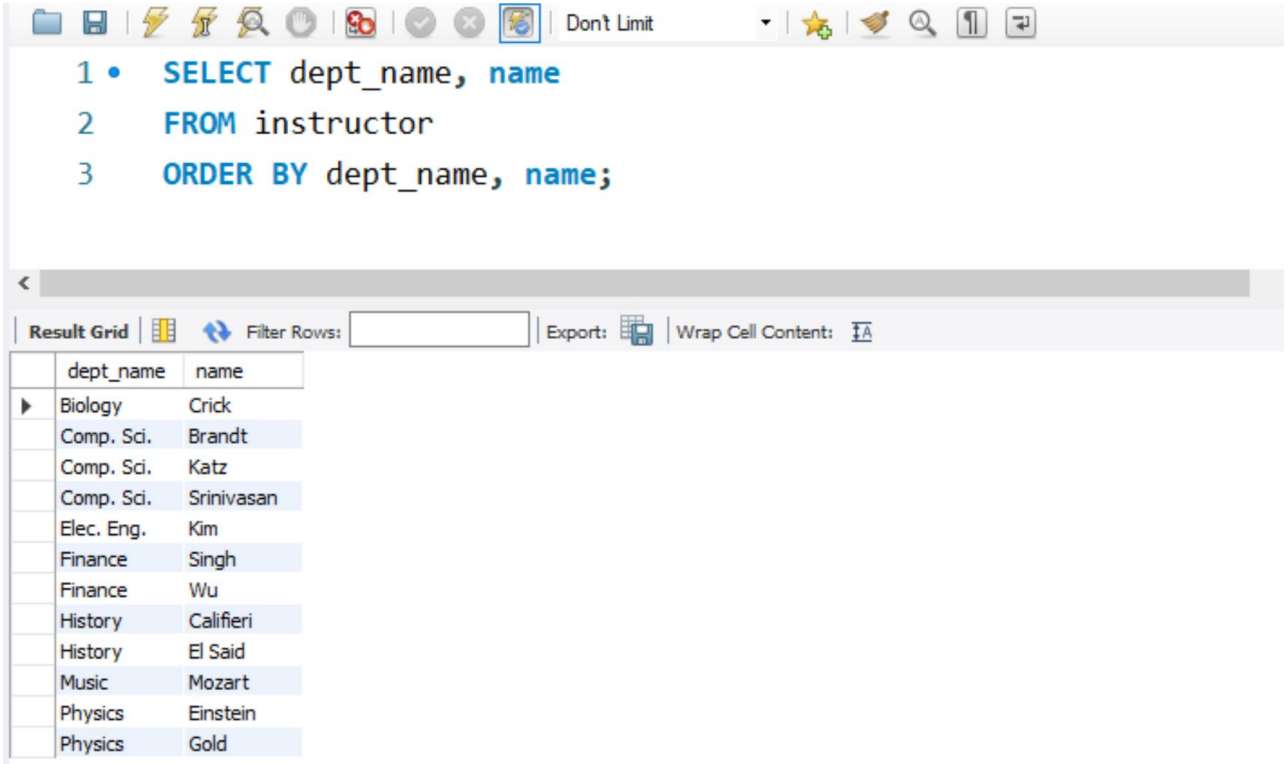
rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
11496	2006-02-14 15:16:03	2047	155	NULL	1	2006-02-15 21:30:53
11541	2006-02-14 15:16:03	2026	335	NULL	1	2006-02-15 21:30:53
11563	2006-02-14 15:16:03	1545	83	NULL	1	2006-02-15 21:30:53
11577	2006-02-14 15:16:03	4106	219	NULL	2	2006-02-15 21:30:53
11593	2006-02-14 15:16:03	817	99	NULL	1	2006-02-15 21:30:53
11611	2006-02-14 15:16:03	1857	192	NULL	2	2006-02-15 21:30:53
11646	2006-02-14 15:16:03	478	11	NULL	2	2006-02-15 21:30:53
11652	2006-02-14 15:16:03	1622	597	NULL	2	2006-02-15 21:30:53
11657	2006-02-14 15:16:03	3043	53	NULL	2	2006-02-15 21:30:53
11672	2006-02-14 15:16:03	3947	521	NULL	2	2006-02-15 21:30:53

The ORDER BY Clause

Northeastern University

- The ORDER BY clause is used to sort the result set

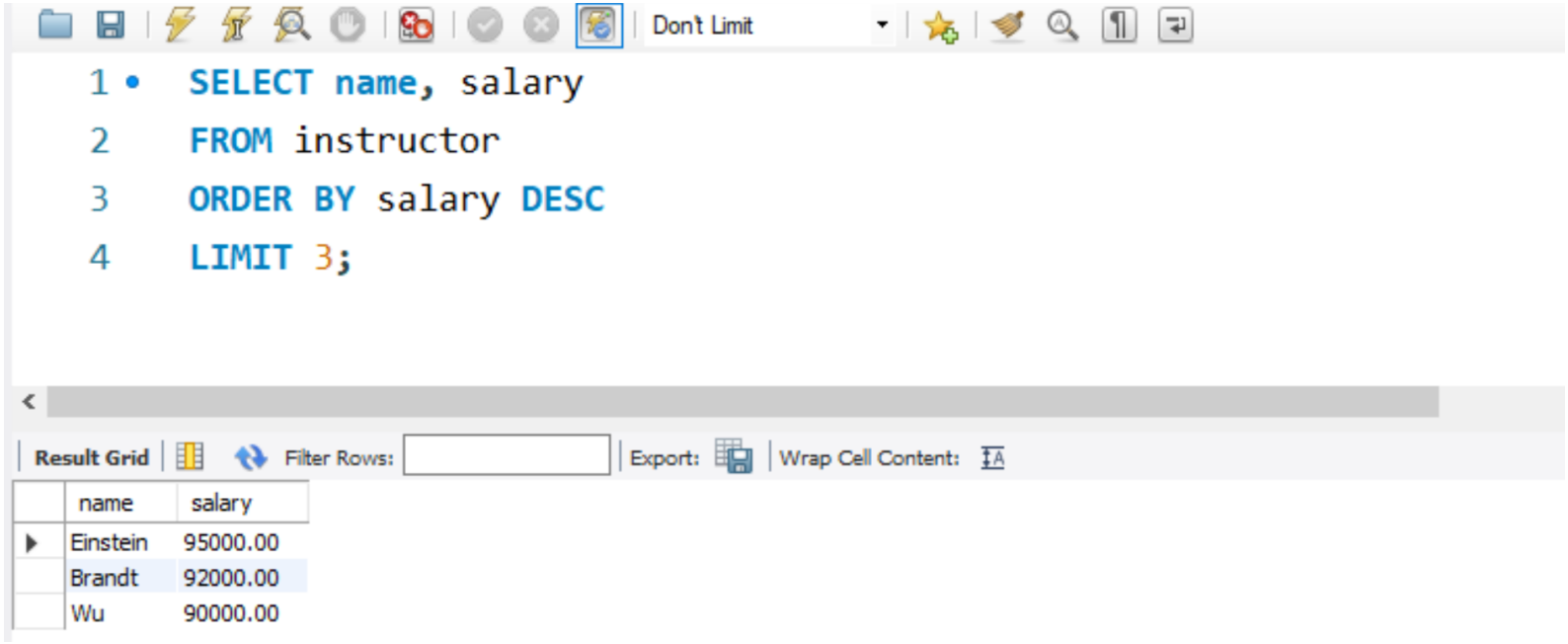
```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 ASC|DESC, column2 ASC|DESC, ...;
```



dept_name	name
Biology	Crick
Comp. Sci.	Brandt
Comp. Sci.	Katz
Comp. Sci.	Srinivasan
Elec. Eng.	Kim
Finance	Singh
Finance	Wu
History	Califieri
History	El Said
Music	Mozart
Physics	Einstein
Physics	Gold

LIMIT

- LIMIT *n* specifies that only the first *n* rows from the result set should be output
- Typically used together with ORDER BY to fetch the top *n* records
- Example: Print the name and salary of the top three earning instructors



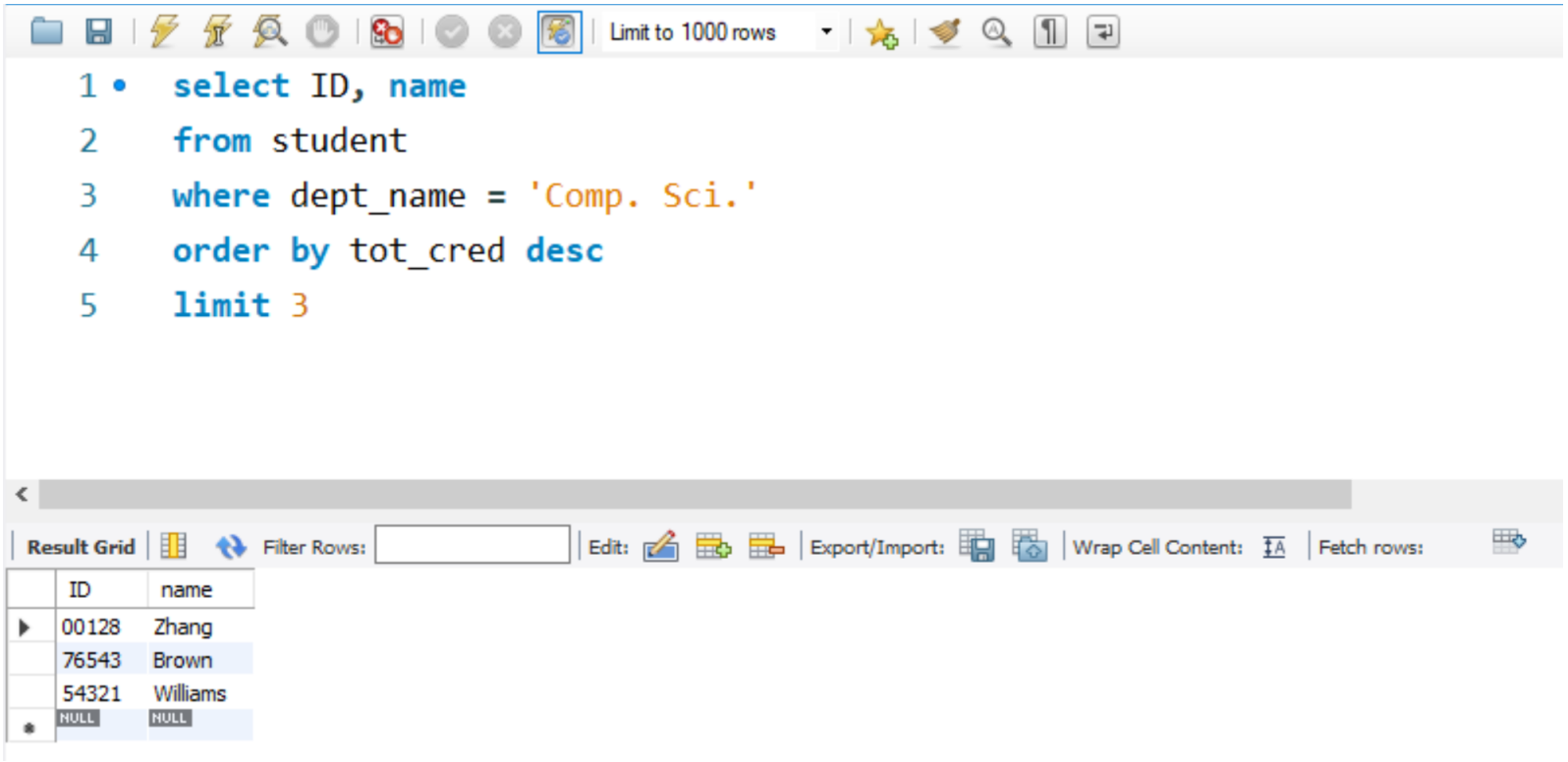
Class Exercise

Northeastern University

- Show the IDs and names of the three students in Comp. Sci. with the highest number of credit points

Solution

Northeastern University



Set Operators

Northeastern University

- You can combine the result sets of two or more queries using set operators
- The three set operations are UNION, INTERSECT and EXCEPT
  - MySQL support only the UNION operator
- When performing set a set operation, the following guidelines must apply:
  - Every select must have the same number of columns
  - The corresponding columns must have similar data types
- The UNION operator syntax:

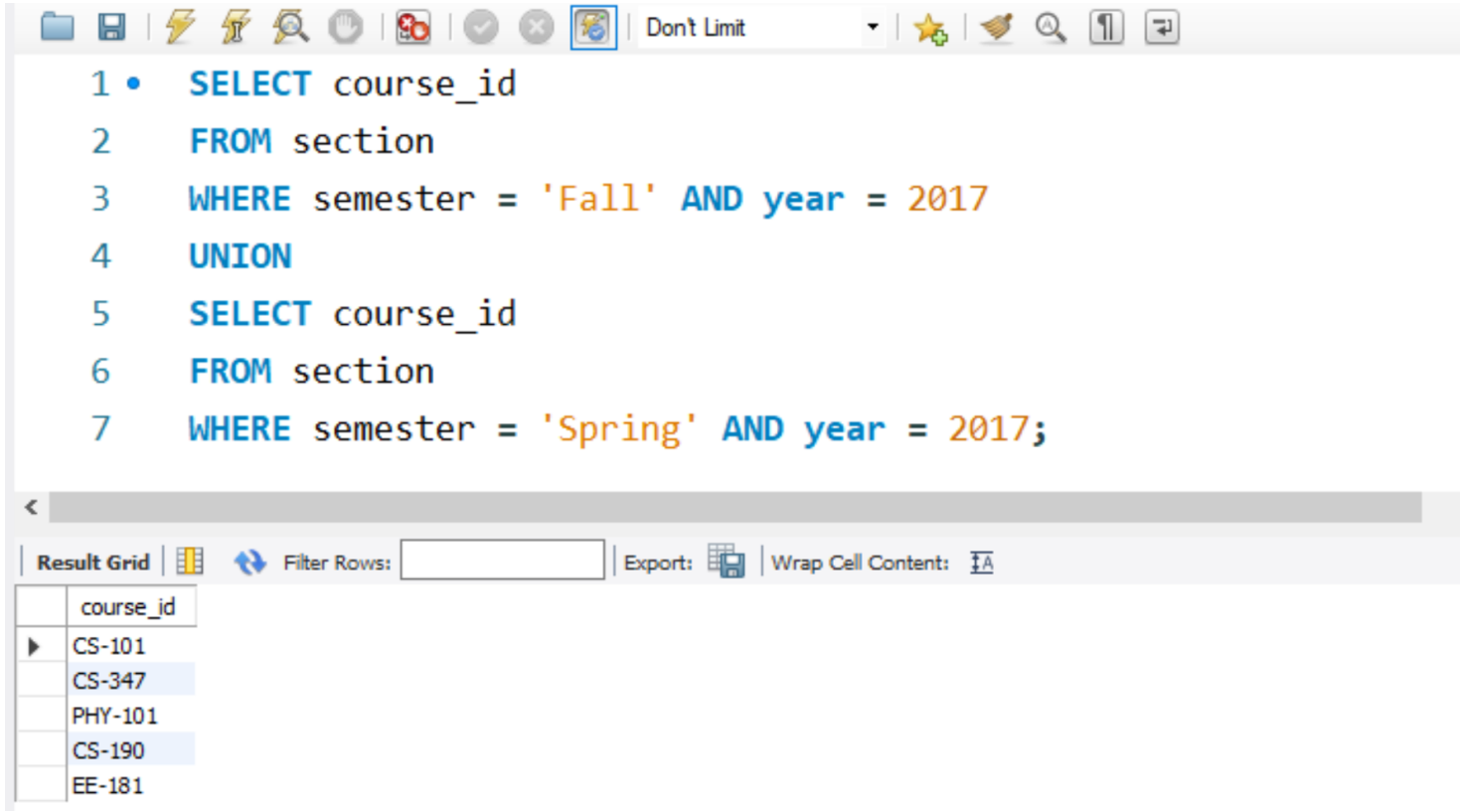
```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

- The UNION operator selects only distinct values by default
  - To allow duplicate values, use UNION ALL

Set Operators

Northeastern University

- Example: Find all the courses taught either in Fall 2017 or Spring 2017



Aggregate Functions

Northeastern University

- Aggregate functions take a set of values of a column and return a single value
- SQL offers 5 built-in aggregate functions:

Function	Description
MAX	Returns the maximum value within a set
MIN	Returns the minimum value within a set
AVG	Returns the average value across a set
SUM	Returns the sum of the values across a set
COUNT	Returns the number of values in a set

Aggregation with Null Values

Northeastern University

- All aggregate functions except COUNT(*) ignore NULL values in their input
  - COUNT(*) returns the number of rows in the table, regardless of NULL values
  - COUNT(column) returns the number of non-null values in the specified column

```
Limit to 1000 rows
1  CREATE TABLE MyTable (
2    my_value SMALLINT
3  );
4
5  INSERT INTO MyTable VALUES (1);
6  INSERT INTO MyTable VALUES (2);
7  INSERT INTO MyTable VALUES (3);
8  INSERT INTO MyTable VALUES (NULL);
9
10 SELECT COUNT(*), COUNT(my_value), SUM(my_value), MAX(my_value), AVG(my_value)
11 FROM MyTable;
```

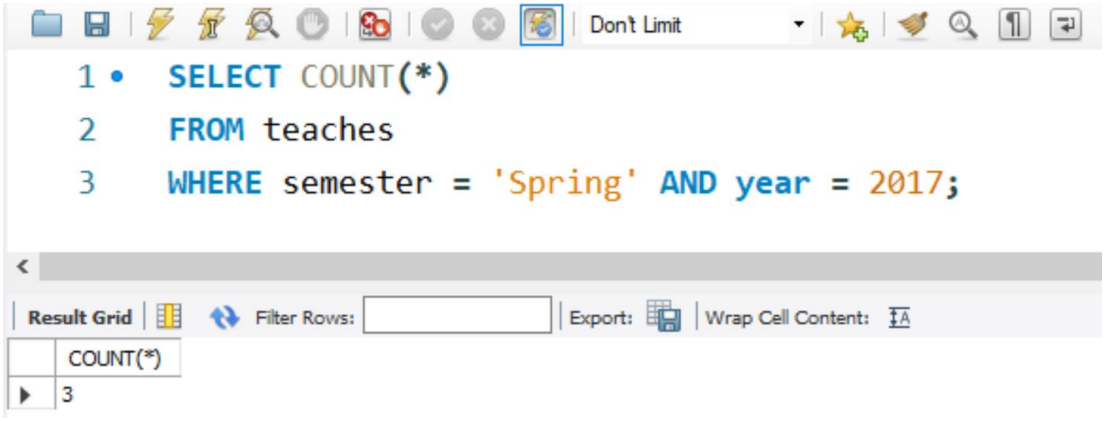
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

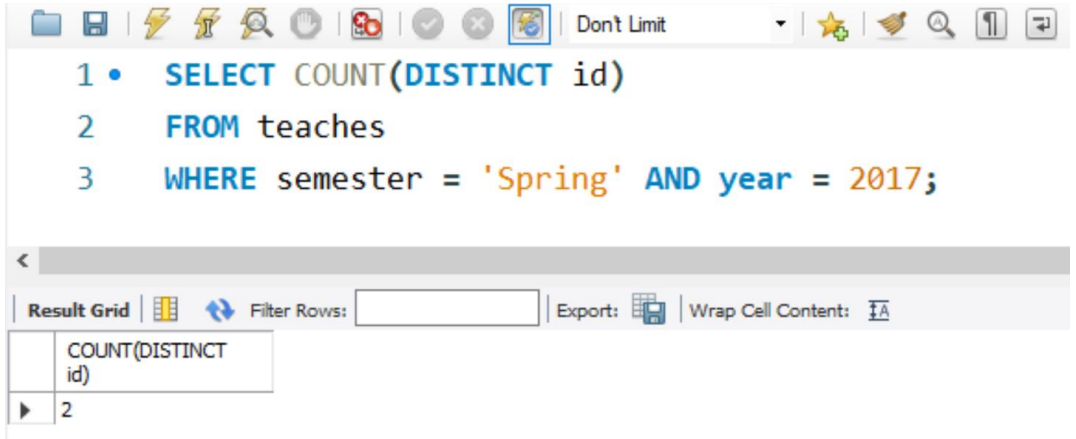
COUNT(*)	COUNT(my_value)	SUM(my_value)	MAX(my_value)	AVG(my_value)
4	3	6	3	2.0000

Counting Distinct Values

Northeastern University

- Sometimes we want to eliminate duplicates before computing an aggregate function
- In this case, we can use DISTINCT inside the call to the function
- Example: Find the total number of instructors who taught a course in Spring 2017
  - Each instructor should only count once, regardless of the number of sections they teach

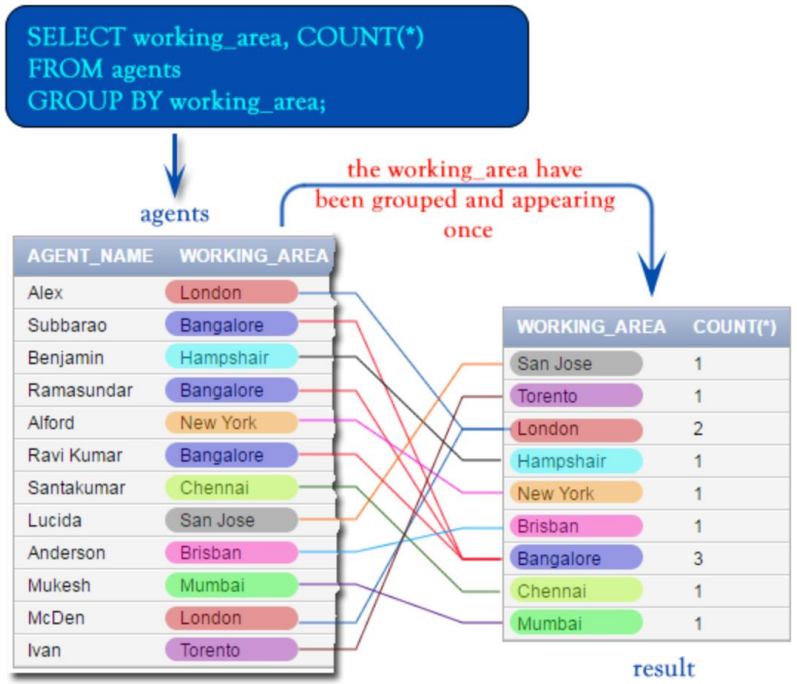




GROUP BY

- The GROUP BY clause groups rows the have the same values in specified column(s)
- Often used with an aggregate function that is applied over all the rows in each group
- The syntax of GROUP BY is:

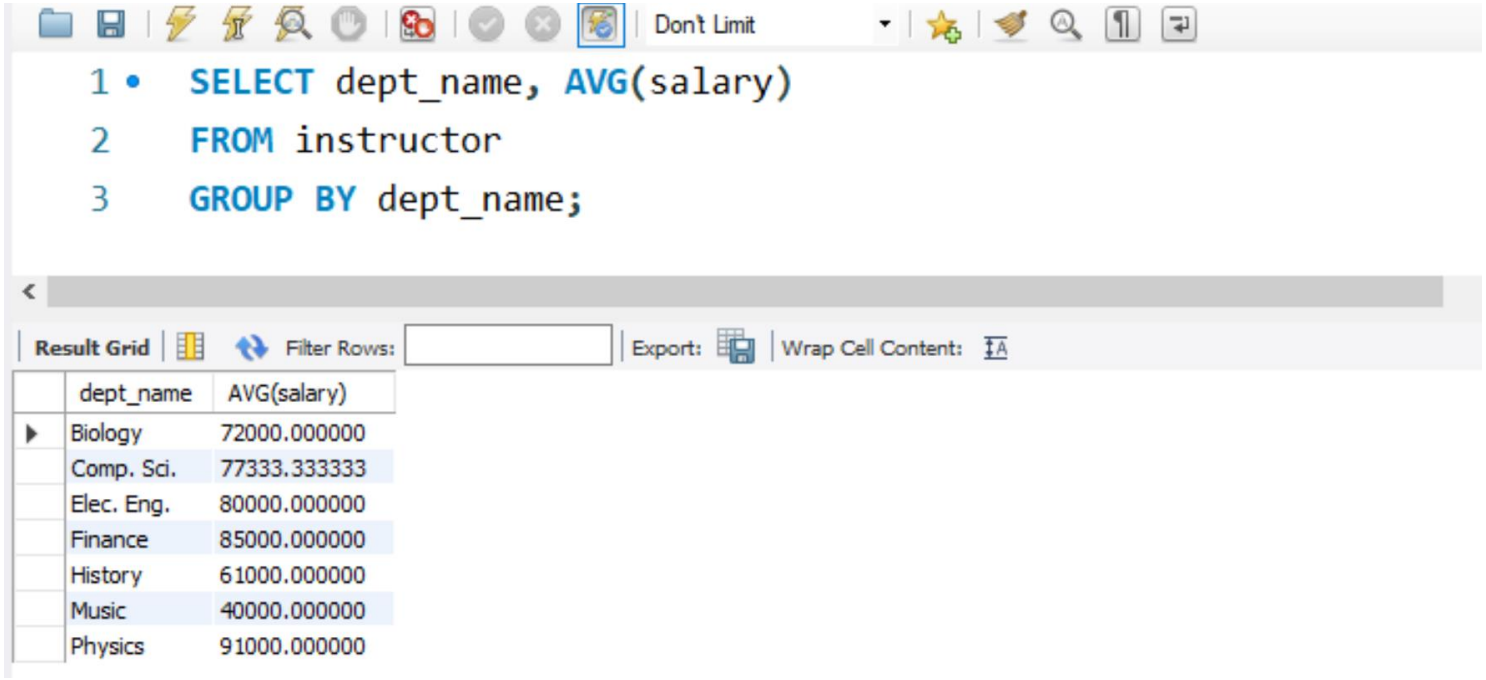
```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s);
```



GROUP BY

Northeastern University

- Example: Find the average salary of instructors in each department



dept_name	AVG(salary)
Biology	72000.000000
Comp. Sci.	77333.333333
Elec. Eng.	80000.000000
Finance	85000.000000
History	61000.000000
Music	40000.000000
Physics	91000.000000

GROUP BY

Northeastern University

- Find the number of different **courses** (not sections) taught by each instructor

```
1 •  SELECT id, COUNT(DISTINCT course_id) AS courses_num
2    FROM teaches
3    GROUP BY id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	courses_num
10101	3
12121	1
15151	1
22222	1
32343	1
45565	2
76766	2
83821	2
98345	1

The Single Value Rule

Northeastern University

- All non-aggregated attributes in the SELECT should also appear in the GROUP BY
  - Most databases (except for MySQL) throw an error if this rule is not followed
- For example, consider the following query:

```
SELECT id, dept_name, AVG(salary)
FROM instructor
GROUP BY dept_name;
```

  - Each instructor in a particular group (defined by *dept_name*) can have a different ID
  - Since only one row is output for each group, there is no unique way of choosing which ID value to output

Grouping by Multiple Columns

Northeastern University

- In some cases, you may want to generate groups based on more than one column
- For each section show the course id, section id, year, semester and students number

```
1 • SELECT course_id, sec_id, semester, year, COUNT(*) AS students_num
2   FROM takes
3   GROUP BY course_id, sec_id, semester, year;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_id	sec_id	semester	year	students_num
BIO-101	1	Summer	2017	1
BIO-301	1	Summer	2018	1
CS-101	1	Fall	2017	6
CS-101	1	Spring	2018	1
CS-190	2	Spring	2017	2
CS-315	1	Spring	2018	2
CS-319	1	Spring	2018	1
CS-319	2	Spring	2018	1
CS-347	1	Fall	2017	2
EE-181	1	Spring	2017	1
FIN-201	1	Spring	2018	1
HIS-351	1	Spring	2018	1
MU-199	1	Spring	2018	1
PHY-101	1	Fall	2017	1

Grouping by Expressions

Northeastern University

- You can also build groups based on values computed by expressions
- Example: For each English letter show how many student names start with that letter

```
1 •  SELECT LEFT(name, 1) AS first_letter, COUNT(*) AS students_num
2    FROM student
3    GROUP BY LEFT(name, 1)
4    ORDER BY first_letter;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_letter	students_num
A	1
B	3
C	1
L	1
P	1
S	3
T	1
W	1
Z	1

The HAVING Clause

Northeastern University

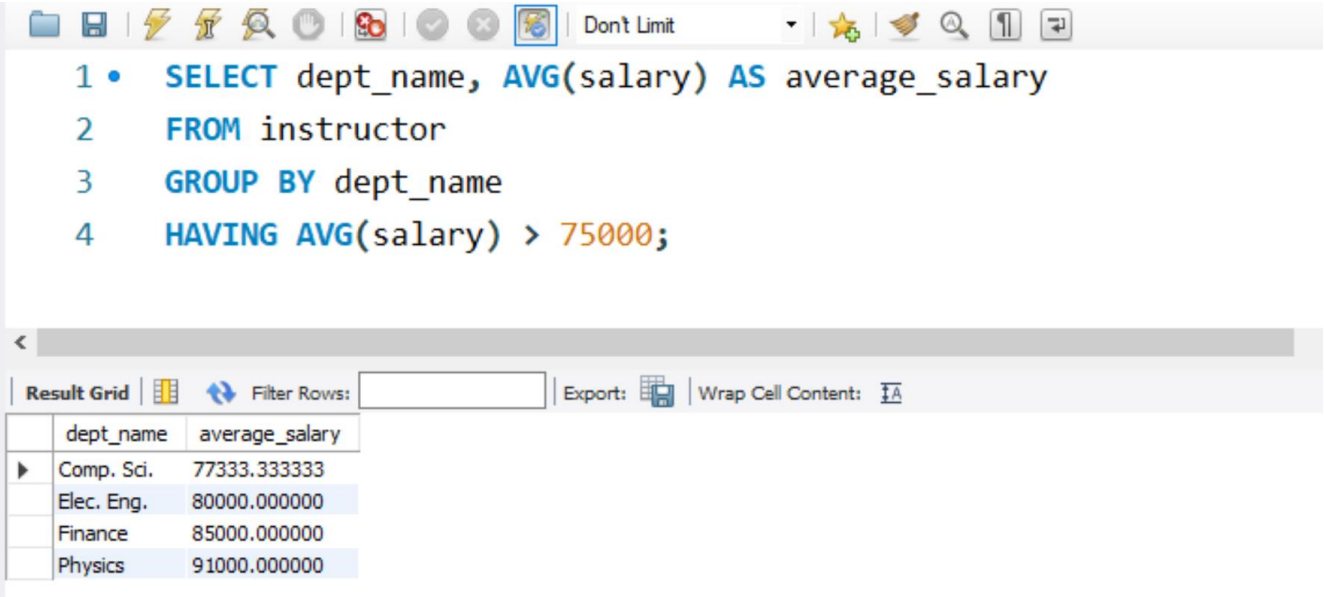
- HAVING allows you to specify conditions that filter the groups created by GROUP BY
  - In contrast to WHERE that defines conditions on the selected rows

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

The HAVING Clause

Northeastern University

- Example: print the names of the departments where the average salary > 75K



```
1 •  SELECT dept_name, AVG(salary) AS average_salary
2    FROM instructor
3    GROUP BY dept_name
4    HAVING AVG(salary) > 75000;
```

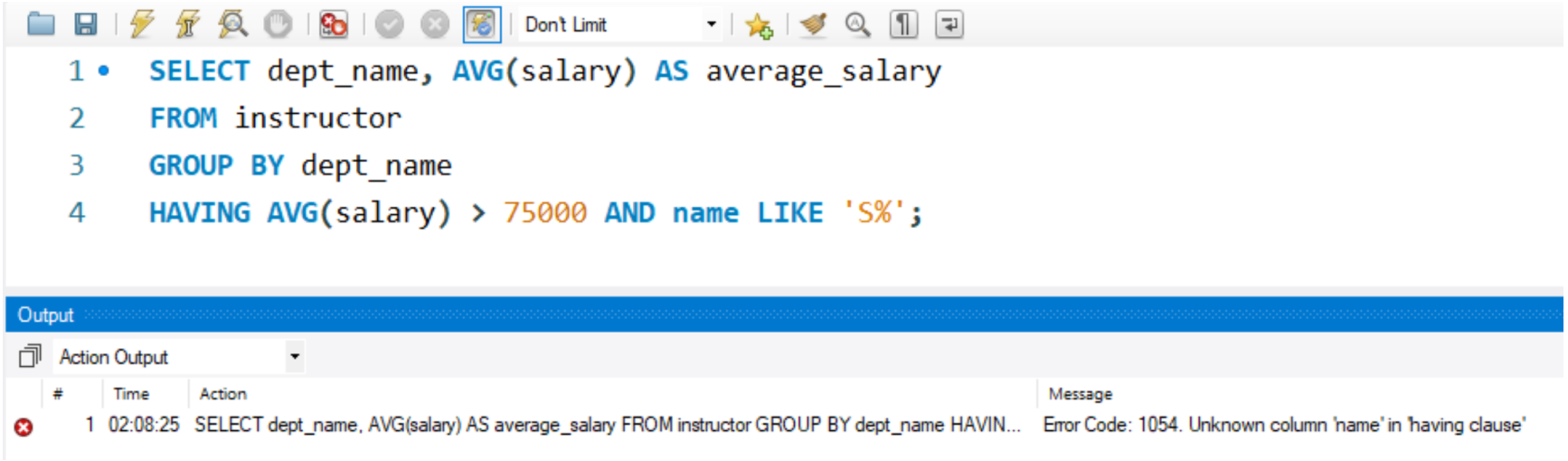
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

dept_name	average_salary
Comp. Sci.	77333.333333
Elec. Eng.	80000.000000
Finance	85000.000000
Physics	91000.000000

The HAVING Clause

Northeastern University

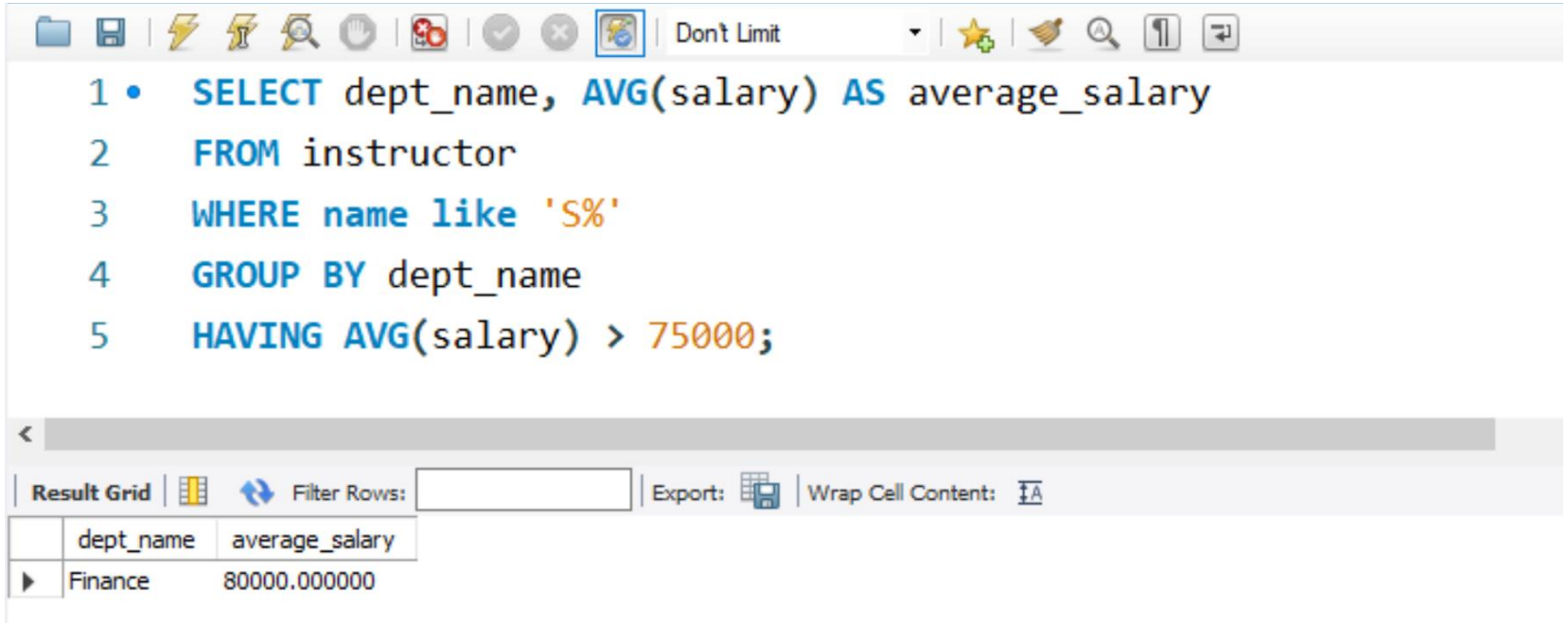
- All non-aggregated columns in HAVING must appear in the GROUP BY clause
- For example, the following SQL statement is invalid:



The HAVING Clause

Northeastern University

- The filter on the name instructor should be written in a WHERE clause instead:



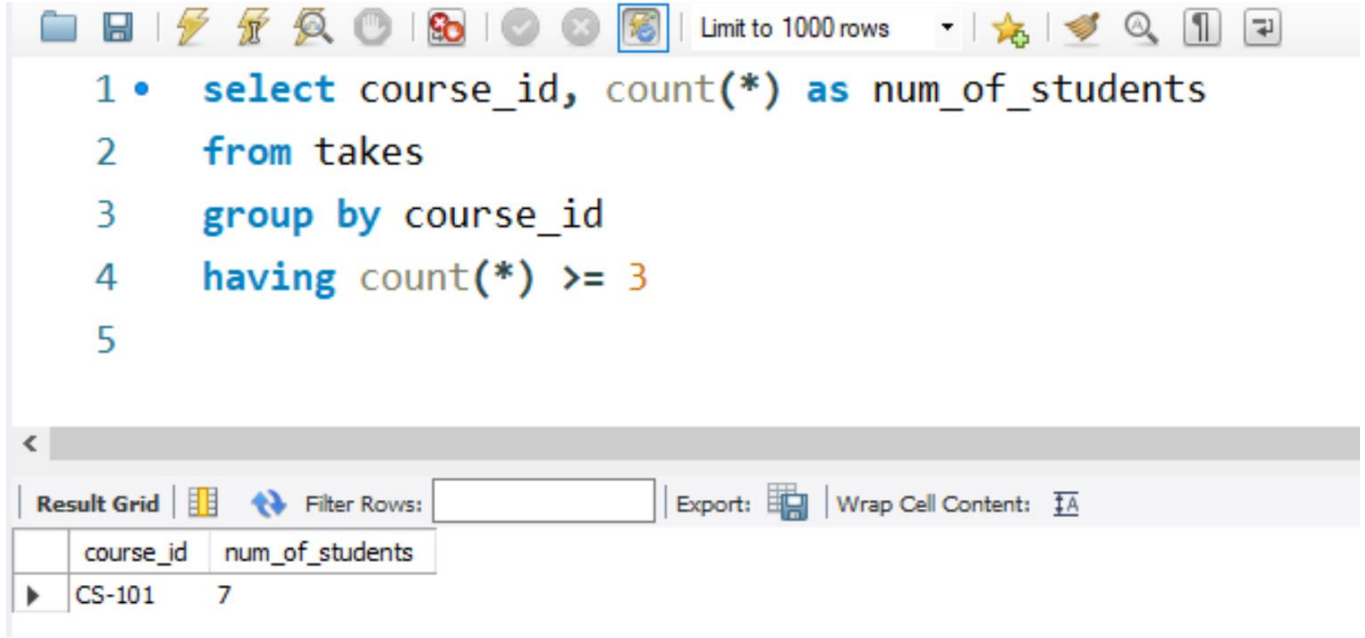
Class Exercise

Northeastern University

- Show the courses that have been taken by at least 3 students
- Show the course ID and number of students

Solution

Northeastern University



SQL Joins

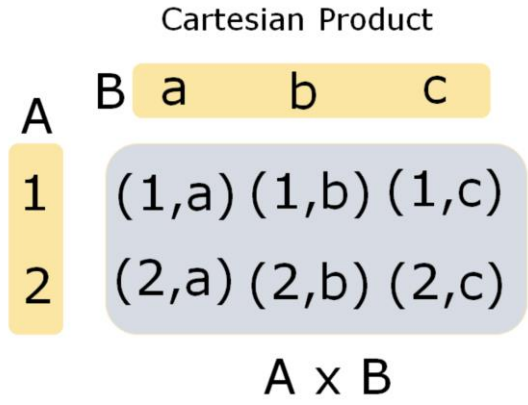
Northeastern University

- A join operation is used to combine rows from two or more tables
  - e.g., find the names of all instructors who teach the course ‘Robotics’
- There are different types of joins:
  - Cross join
  - Inner join
  - Self join
  - Outer join

Cross Join

Northeastern University

- A cross join returns the Cartesian product of rows from the tables in the join
  - i.e., it combines each row from the first table with each row from the second table



- Two ways to define a cross join:
  - Use the JOIN keyword between the table names

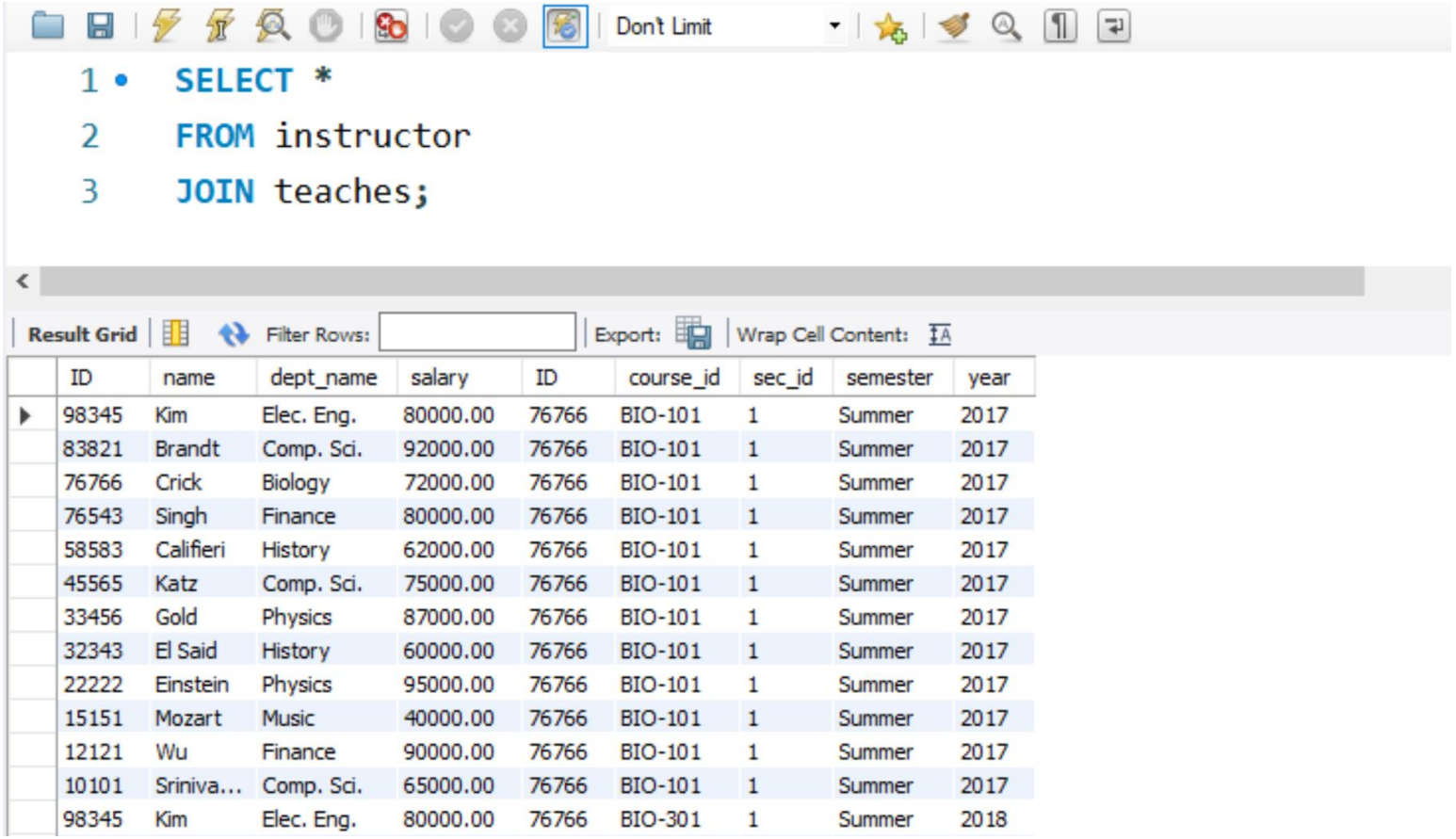
```
SELECT column_name(s)
FROM table1
JOIN table2;
```

  - List the tables with a comma between them (older syntax)

```
SELECT column_name(s)
FROM table1, table2;
```

Cross Join

- Example: cross join the instructor and the teaches tables



```
1 • SELECT *
2   FROM instructor
3   JOIN teaches;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	dept_name	salary	ID	course_id	sec_id	semester	year
98345	Kim	Elec. Eng.	80000.00	76766	BIO-101	1	Summer	2017
83821	Brandt	Comp. Sci.	92000.00	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000.00	76766	BIO-101	1	Summer	2017
76543	Singh	Finance	80000.00	76766	BIO-101	1	Summer	2017
58583	Califieri	History	62000.00	76766	BIO-101	1	Summer	2017
45565	Katz	Comp. Sci.	75000.00	76766	BIO-101	1	Summer	2017
33456	Gold	Physics	87000.00	76766	BIO-101	1	Summer	2017
32343	El Said	History	60000.00	76766	BIO-101	1	Summer	2017
22222	Einstein	Physics	95000.00	76766	BIO-101	1	Summer	2017
15151	Mozart	Music	40000.00	76766	BIO-101	1	Summer	2017
12121	Wu	Finance	90000.00	76766	BIO-101	1	Summer	2017
10101	Sriniva...	Comp. Sci.	65000.00	76766	BIO-101	1	Summer	2017
98345	Kim	Elec. Eng.	80000.00	76766	BIO-301	1	Summer	2018

Inner Join

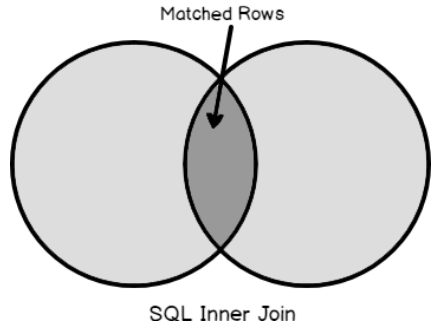
Northeastern University

- An inner join selects only rows that have matching values in both tables
- Two ways to define an inner join:
  - Using the JOIN keyword with ON clause

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

    - The keyword INNER is optional, since the default join type is inner
  - List the tables with a comma between and use WHERE to define the join condition

```
SELECT column_name(s)
FROM table1, table2
WHERE table1.column_name = table2.column_name;
```

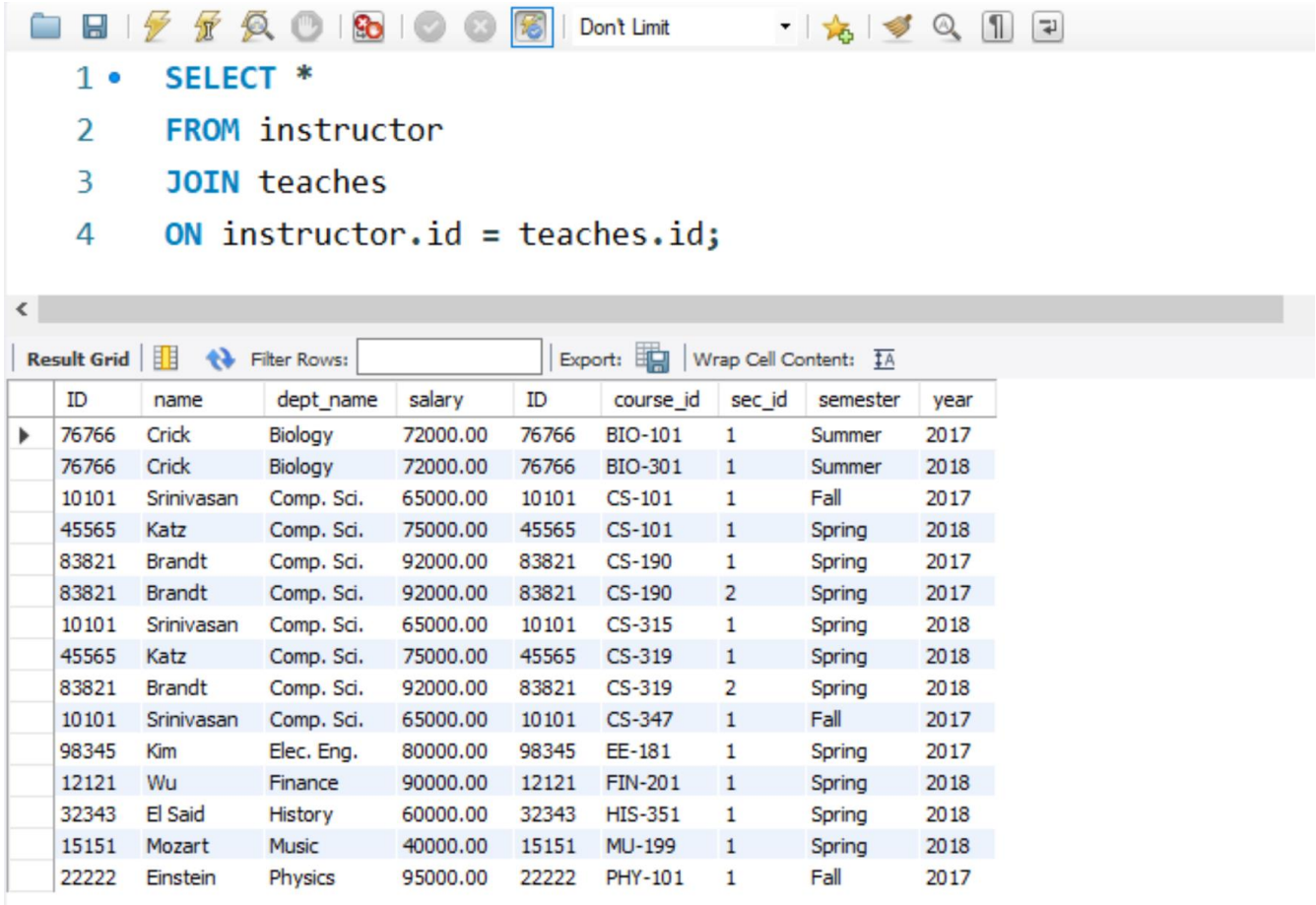


- The first syntax should be preferred, because it encourages separation of concerns

Inner Join

Northeastern University

- Example: show all the instructors and the courses that they have taught



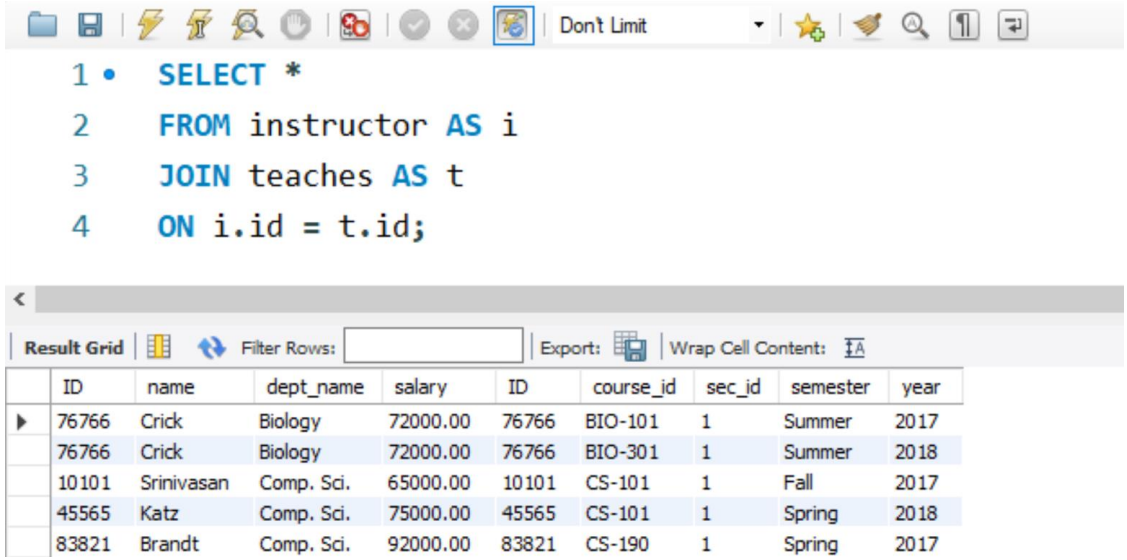
```
1 •  SELECT *
2    FROM instructor
3    JOIN teaches
4    ON instructor.id = teaches.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	dept_name	salary	ID	course_id	sec_id	semester	year
76766	Crick	Biology	72000.00	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000.00	76766	BIO-301	1	Summer	2018
10101	Srinivasan	Comp. Sci.	65000.00	10101	CS-101	1	Fall	2017
45565	Katz	Comp. Sci.	75000.00	45565	CS-101	1	Spring	2018
83821	Brandt	Comp. Sci.	92000.00	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000.00	83821	CS-190	2	Spring	2017
10101	Srinivasan	Comp. Sci.	65000.00	10101	CS-315	1	Spring	2018
45565	Katz	Comp. Sci.	75000.00	45565	CS-319	1	Spring	2018
83821	Brandt	Comp. Sci.	92000.00	83821	CS-319	2	Spring	2018
10101	Srinivasan	Comp. Sci.	65000.00	10101	CS-347	1	Fall	2017
98345	Kim	Elec. Eng.	80000.00	98345	EE-181	1	Spring	2017
12121	Wu	Finance	90000.00	12121	FIN-201	1	Spring	2018
32343	El Said	History	60000.00	32343	HIS-351	1	Spring	2018
15151	Mozart	Music	40000.00	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000.00	22222	PHY-101	1	Fall	2017

Renaming Tables

- Aliases are often used in JOIN statements to rename the tables



```
1 •  SELECT *
2    FROM instructor AS i
3    JOIN teaches AS t
4    ON i.id = t.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	dept_name	salary	ID	course_id	sec_id	semester	year
76766	Crick	Biology	72000.00	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000.00	76766	BIO-301	1	Summer	2018
10101	Srinivasan	Comp. Sci.	65000.00	10101	CS-101	1	Fall	2017
45565	Katz	Comp. Sci.	75000.00	45565	CS-101	1	Spring	2018
83821	Brandt	Comp. Sci.	92000.00	83821	CS-190	1	Spring	2017

- Conventions:
  - The alias name should be the first letter of each word in the table's name
  - If there is already an alias with the same name then append a number (e.g., t1, t2)
  - Always include the AS keyword – makes it easier to read the query

Joining More than Two Tables

Northeastern University

- Example: Print the names of the instructors and names of all the students they have taught (make sure there are no duplicates in the result)

```
1 •  SELECT DISTINCT i.name AS instructor_name, s.name AS student_name
2    FROM instructor AS i
3    JOIN teaches
4    ON i.id = teaches.id
5    JOIN takes
6    ON teaches.course_id = takes.course_id
7      AND teaches.sec_id = takes.sec_id
8      AND teaches.semester = takes.semester
9      AND teaches.year = takes.year
10   JOIN student AS s
11   ON s.id = takes.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

instructor_name	student_name
Crick	Tanaka
Srinivasan	Zhang
Srinivasan	Shankar
Srinivasan	Levy

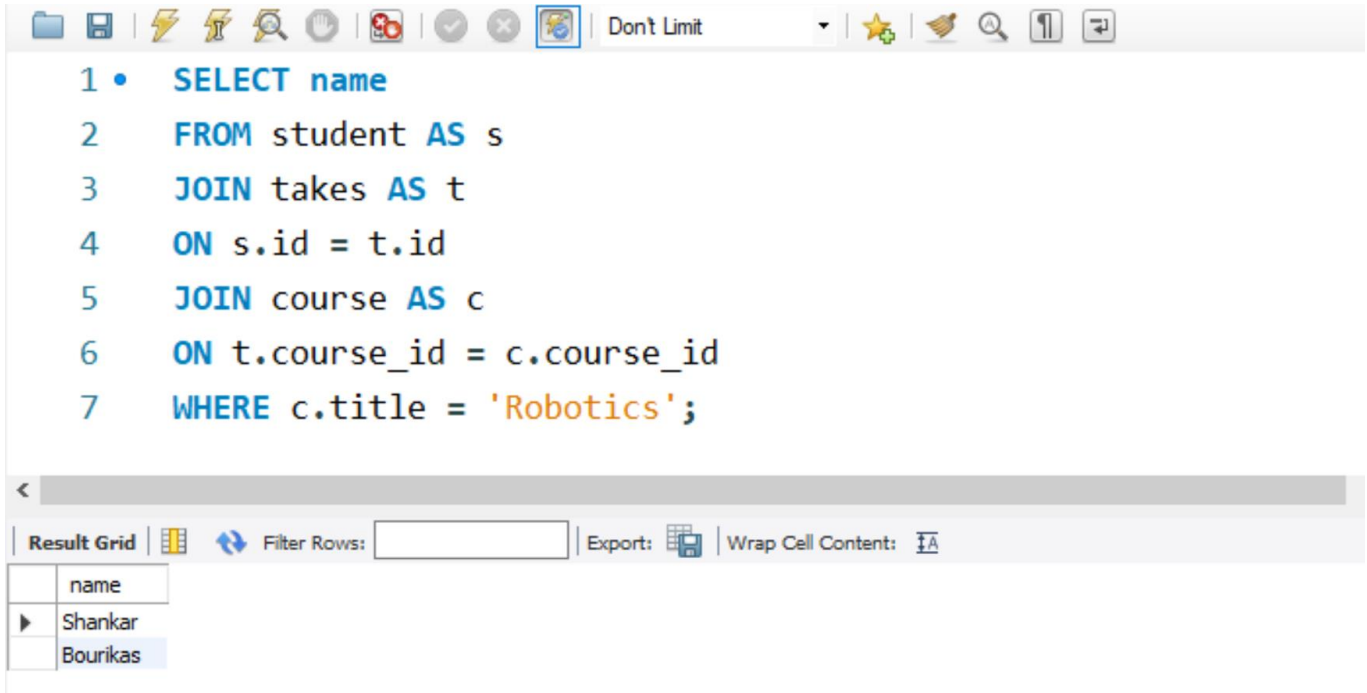
Class Exercise

Northeastern University

- Find the names of all the students that took the course 'Robotics'

Solution

Northeastern University



Self-Join

Northeastern University

- A self-join is a join of a table with itself
- It allows you to compare rows within the same table
- Example: Find all the instructors whose salary is higher than Katz's salary

```
1 •  SELECT i1.name, i1.salary
2    FROM instructor AS i1
3    JOIN instructor AS i2
4    ON i1.salary > i2.salary
5    WHERE i2.name = 'Katz';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	salary
Wu	90000.00
Einstein	95000.00
Gold	87000.00
Singh	80000.00
Brandt	92000.00
Kim	80000.00

Outer Join

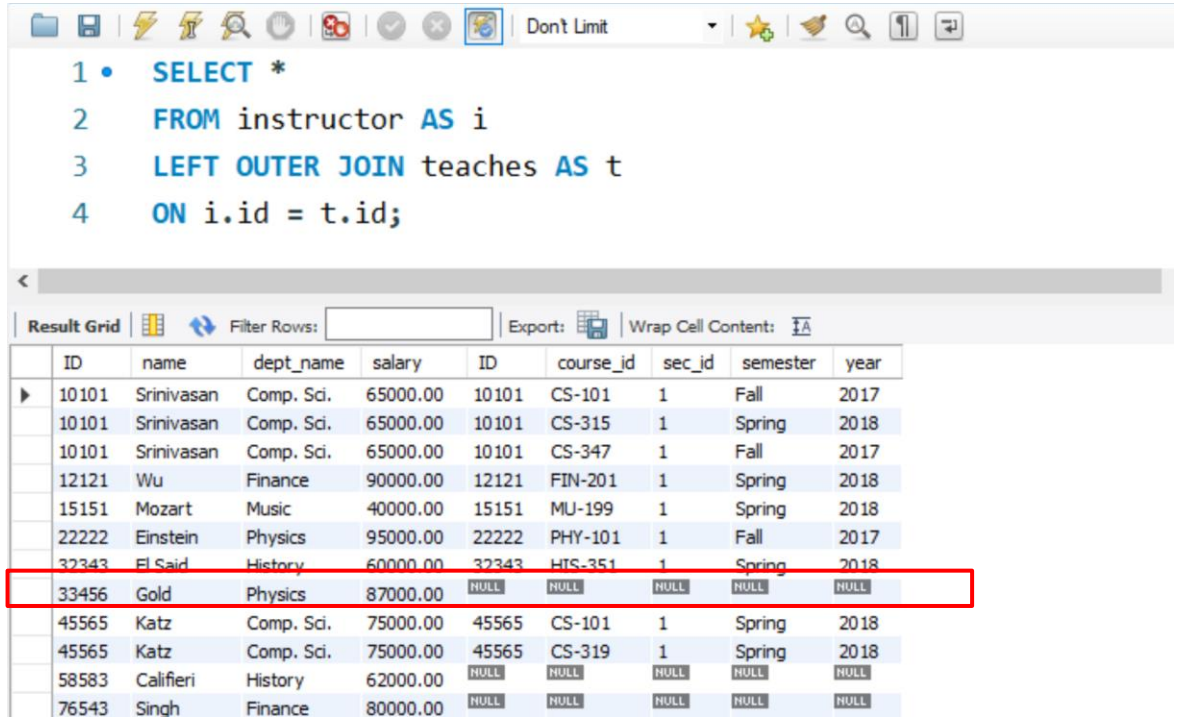
Northeastern University

- Outer join allows you to include unmatched rows from one or both of the tables
  - The unmatched rows will have NULL values for the attributes coming from the other table
- There are three forms of outer join
  - **Left outer join** includes unmatched rows from the left table
  - **Right outer join** includes unmatched rows from the right table
  - **Full outer join** includes unmatched rows from both tables
- The syntax for outer join:

```
SELECT column_name(s)
FROM table1
[LEFT/RIGHT/FULL] OUTER JOIN table2
ON table1.column_name = table2.column_name;
```

Left Outer Join

- For example, let's say we want to show all instructors and the courses they taught
- An inner join won't show instructors that have not taught any course
- If we also want to get these instructors in the result, we can use a left outer join:



```
1 • SELECT *
2   FROM instructor AS i
3   LEFT OUTER JOIN teaches AS t
4   ON i.id = t.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	dept_name	salary	ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000.00	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000.00	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000.00	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000.00	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000.00	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000.00	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000.00	32343	HIS-351	1	Spring	2018
33456	Gold	Physics	87000.00	NULL	NULL	NULL	NULL	NULL
45565	Katz	Comp. Sci.	75000.00	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000.00	45565	CS-319	1	Spring	2018
58583	Califieri	History	62000.00	NULL	NULL	NULL	NULL	NULL
76543	Singh	Finance	80000.00	NULL	NULL	NULL	NULL	NULL

Right Outer Join

- We can write a similar query using a right outer join with the tables order reversed:

```
1 •  SELECT *
2    FROM teaches AS t
3    RIGHT OUTER JOIN instructor AS i
4    ON i.id = t.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	course_id	sec_id	semester	year	ID	name	dept_name	salary
10101	CS-101	1	Fall	2017	10101	Srinivasan	Comp. Sci.	65000.00
10101	CS-315	1	Spring	2018	10101	Srinivasan	Comp. Sci.	65000.00
10101	CS-347	1	Fall	2017	10101	Srinivasan	Comp. Sci.	65000.00
12121	FIN-201	1	Spring	2018	12121	Wu	Finance	90000.00
15151	MU-199	1	Spring	2018	15151	Mozart	Music	40000.00
22222	PHY-101	1	Fall	2017	22222	Einstein	Physics	95000.00
32343	HIS-351	1	Spring	2018	32343	El Said	History	60000.00
NULL	NULL	NULL	NULL	NULL	33456	Gold	Physics	87000.00
45565	CS-101	1	Spring	2018	45565	Katz	Comp. Sci.	75000.00
45565	CS-319	1	Spring	2018	45565	Katz	Comp. Sci.	75000.00
NULL	NULL	NULL	NULL	NULL	58583	Califieri	History	62000.00
NULL	NULL	NULL	NULL	NULL	76543	Singh	Finance	80000.00

Full Outer Join

Northeastern University

- The full outer join is a combination of the left and right outer join types
- Example: display a list of all instructors and their departments
  - All departments must be displayed, even if they have no instructor assigned to them
  - All instructors must be displayed, even if they don't belong to any department

```
SELECT *
FROM instructor AS i
FULL OUTER JOIN department AS d
ON i.dept_name = d.dept_name;
```

- Not supported in MySQL
  - Can be emulated using LEFT OUTER JOIN + UNION + RIGHT OUTER JOIN

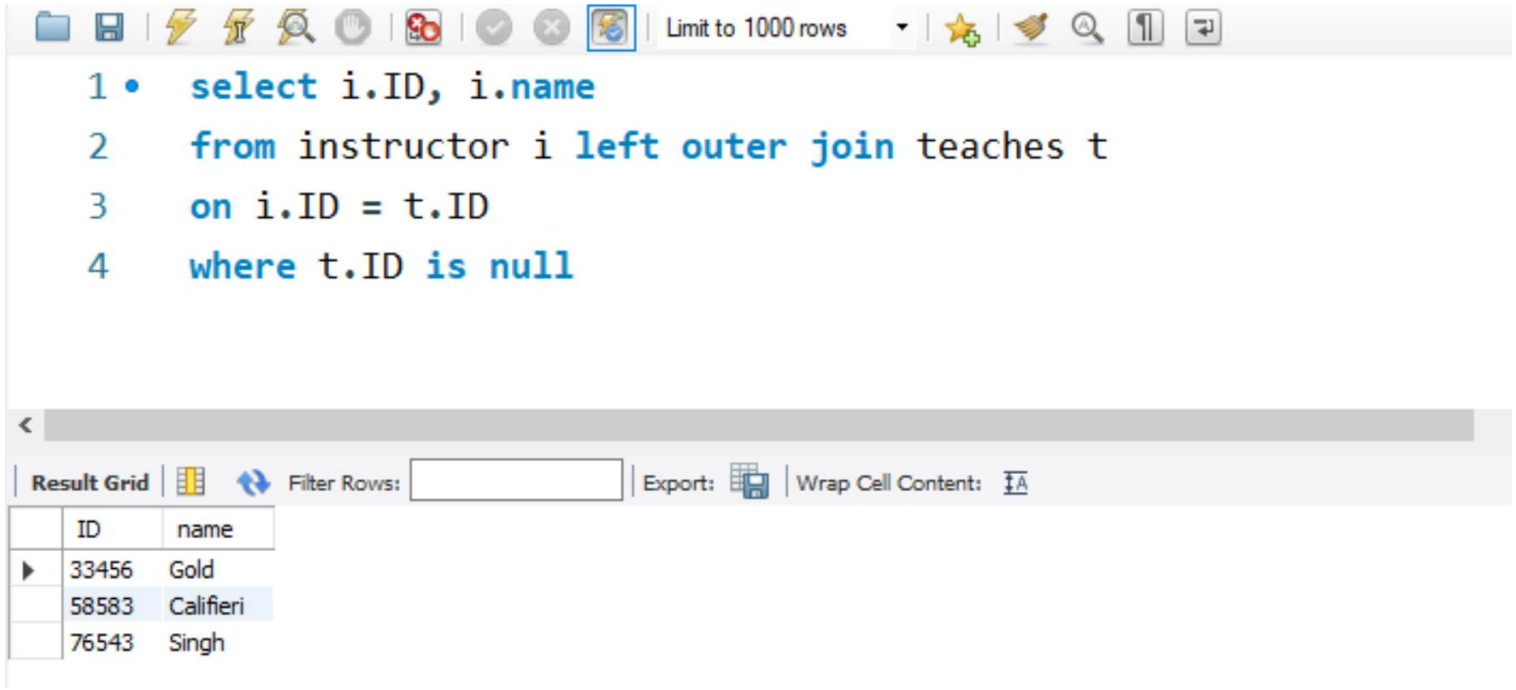
Class Exercise

Northeastern University

- Find the ID and names of instructors who have not taught any course

Solution

Northeastern University



Summary: Join Types

Northeastern University