CS-401 MODERN PROGRAMMING PRACTICES

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
OOAD / RDBMS / OOP	**AM: Lesson 1:** *The OO Paradigm for Building Software Solutions* **PM: Lab 1**	**AM: Lesson 2:** *Associations among Objects and Classes* [Lab 1 due 10 AM] **PM: Lab 1 solutions, Lab 2**	**AM: Lesson 3:** *Inheritance and Composition* [Lab 2 due 10 AM] **PM: Lab 2 solutions, Lab 3**	**AM: Lesson 4**: *Interaction Diagrams* [Lab 3 due 10 AM] **PM: Lab 3 solutions, Lab 4**	**AM: Lesson 5**: *Inheritance and Abstraction* [Lab 4 due 10 AM] **PM: Lab 4 solutions, Lab 5**	**AM: Lesson 6:** *Relational Model, View & Normalization* [Lab 5 due 10 AM] **Lab 5 solutions**
	AM: Lesson 7: *SQL (DML&DDL)* **PM: Lab 6, 7**	**AM: Lesson 8**: *Index, SQL Injection, JDBC application & Intro to Maven* [Lab 6 due 10 AM] [Lab 7 due 5 PM] **PM: Lab 6 solutions, Lab 8**	**AM:** Review for Midterm exam **Lab 7 & 8 solutions** **PM:** *Study for Midterm*	**MIDTERM EXAM**	**AM: Lesson 9**: *Interfaces in Java 8 and the Object Superclass* **PM: Lab 9**	**AM: Lesson 10**: *Functional Programming in Java* [Lab 9 due 10 AM] **Lab 9 solutions**
OOP	**AM: Lesson 11**: ***The Stream API*** **PM: Lab 10**	**AM: Lesson 11**: ***The Stream API*** [Lab 10 due 10 AM] **PM: Lab 10 solutions, Lab**	**AM: Lesson 12** *Best Programming Practices with Java 8* [Lab 11 due 10 AM] **PM: Lab 11 solutions, Lab**	**AM: Lesson 13** *Generic Programming* **PM: Lab 12 solutions, Lab 13**	**AM**: Review for Final exam **Lab 13 solutions**	**Final Exam**
	AM: Java Swing **PM: Course Project**	**Course Project**	**Course Project**	**AM:** *Project Presentation – Connect whole with parts*		

Lesson 7

DDL & DML

Dr. Bright Gee Varghese

Wholeness

SQL is a non-procedural language that can be used by professionals and non-professionals alike. It is both a formal and de facto standard language for defining and manipulating relational databases.

Science & Technology of Consciousness: TM is a simple, effortless mental technique that can be used by anyone, no matter what their lifestyle. It promotes spontaneous fulfillment of desires, by bringing the desires of the individual into accord with Natural Law, without the individual having to know the underlying mechanism.

Outline

- Introduction to SQL commands
- DDL
- DML
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

Outline

- Introduction to SQL commands
- DDL
- DML
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

Introduction to SQL commands

- SQL (Structured Query Language) is a standard language used to interact with relational databases. It is used to store, retrieve, manipulate, and manage data efficiently.

Why Use SQL?

Data Retrieval: Extract data from databases using queries.

Data Manipulation: Insert, update, and delete records.

Database Management: Define schemas, create and modify tables.

Data Control: Manage user access and permissions.

SQL Commands

SQL commands are divided into five main groups based on their use.

Type	Full Form	What it does	Example
DDL	Data Definition Language	Makes or changes tables	`CREATE`, `ALTER`, `DROP`
DML	Data Manipulation Language	Adds, changes, and deletes data	`INSERT`, `UPDATE`, `DELETE`
DQL	Data Query Language	Asks questions to get data	`SELECT`
DCL	Data Control Language	Controls who can access data	`GRANT`, `REVOKE`
TCL	Transaction Control Language	Manages changes as one group	`COMMIT`, `ROLLBACK`

SQL Identifier

- In SQL, an identifier is a name you give to a database object, such as a table, column, schema, index, or constraint.
  - The identifier is a string of characters from some set.
  - The string is at most 128 characters long.
  - The standard set of characters consists of capital letters (A,B,..), small letters (a,b,...), digits (0,1,...) and the underscore character (_).
  - An identifier starts with a letter.
  - An identifier contains no spaces (allowed by Access but not by standard SQL).

Outline

- Introduction to SQL commands
- DDL
- DML
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

DDL (Data Definition Language)

- It is used to define a database schema or modify an existing one.
- It cannot be used to manipulate data.
- It consists of the SQL commands that can be used for defining, altering, and deleting database structures such as tables, indexes, and schemas.

The main SQL DDL statements are:		
CREATE SCHEMA		DROP SCHEMA
CREATE DOMAIN	ALTER DOMAIN	DROP DOMAIN
CREATE TABLE	ALTER TABLE	DROP TABLE
CREATE VIEW		DROP VIEW
CREATE INDEX		DROP INDEX

DDL (Data Definition Language)

Command	**Description**	**Syntax**
CREATE	Create database or its objects (table, index, function, views, store procedure, and triggers)	**CREATE** TABLE table_name (column1 data_type, column2 data_type, ...);
DROP	Delete objects from the database	**DROP** TABLE table_name;
ALTER	Alter the structure of the database	**ALTER** TABLE table_name ADD COLUMN column_name data_type;
TRUNCATE	Remove all records from a table, including all spaces allocated for the records are removed	**TRUNCATE** TABLE table_name;
RENAME	Rename an object existing in the database	**RENAME** TABLE old_table_name TO new_table_name;

CREATE Command

Creating a Database

- The CREATE DATABASE statement is used to create a new SQL database.

- Syntax
  - CREATE DATABASE databasename;
- Example
  - CREATE DATABASE libraryDB;

DDL Commands
CREATE
DROP
ALTER
TRUNCATE
RENAME

CREATE Command

Creating a Database

```
 MySQL  localhost:3306 ssl  SQL  CREATE DATABASE miu_db;
Query OK, 1 row affected (0.0015 sec)
 MySQL  localhost:3306 ssl  SQL  CREATE USER 'app_user'@'localhost' IDENTIFIED BY 'Pwde3';
Query OK, 0 rows affected (0.0041 sec)
 MySQL  localhost:3306 ssl  SQL  GRANT ALL PRIVILEGES ON miu_db.* TO 'app_user'@'localhost';
Query OK, 0 rows affected (0.0012 sec)
 MySQL  localhost:3306 ssl  SQL  SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| admission_db       |
| bank_db            |
| ecommerce_db       |
| information_schema |
| miu_db             |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
8 rows in set (0.0007 sec)
 MySQL  localhost:3306 ssl  SQL
```

```
GRANT ALL PRIVILEGES ON miu_db.* TO 'app_user'@'192.168.10.25';
```

CREATE Command
Creating a Table

- The CREATE TABLE statement is used to create a new table in a database.

- Syntax

```
CREATE TABLE table_name (
 column1 datatype [constraints],
 column2 datatype [constraints],
 column3 datatype [constraints],
 ...
);
```

```sql
CREATE TABLE Employees(
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(50) UNIQUE,
    salary DECIMAL(10, 2) CHECK(salary > 0),
    gender CHAR(1) CHECK(gender IN ('M', 'F')),
    hire_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

CREATE Command

Creating a Table

```
MySQL  localhost:3306 ssl  SQL  USE miu_db;
Default schema set to `miu_db`.
Fetching global names, object names from `miu_db` for auto-completion... Press ^C to stop.
MySQL  localhost:3306 ssl  miu_db  SQL  CREATE TABLE Employees(
employee_id INT AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(50) NOT NULL,
last_name VARCHAR(50) NOT NULL,
email VARCHAR(50) UNIQUE,
salary DECIMAL(10, 2) CHECK(salary > 0),
gender CHAR(1) CHECK(gender IN ('M', 'F')),
hire_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
Query OK, 0 rows affected (0.0340 sec)
MySQL  localhost:3306 ssl  miu_db  SQL  SHOW TABLES;
+-----------------+
| Tables_in_miu_db |
+-----------------+
| Employees       |
+-----------------+
1 row in set (0.0014 sec)
MySQL  localhost:3306 ssl  miu_db  SQL
```

Data Types and Constraints

Component	Description
INT	Integer data type
VARCHAR(n)	Variable-length string (max n characters)
DECIMAL(p,s)	Fixed-point number with p digits and s decimal places
DATE	Stores date values (YYYY-MM-DD)
TIMESTAMP	Returns a datetime value based on a date or datetime value
PRIMARY KEY	Uniquely identifies each row
NOT NULL	Ensures column cannot have NULL values
CHECK(condition)	Enforces condition on column values
DEFAULT	Assigns a default value if none is provided

ALTER Command

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

DDL Commands
CREATE
DROP
ALTER
TRUNCATE
RENAME

ALTER Command

- Add a column in a table:
  - Syntax
    - ALTER TABLE table_name ADD column_name datatype;
  - Example
    - ALTER TABLE Employees ADD phone_number VARCHAR(15);

- Delete a column in a table:
  - Syntax
    - ALTER TABLE table_name DROP COLUMN column_name;
  - Example
    - ALTER TABLE Employees DROP phone_number;

ALTER Command
DDL

- Rename a column in a table:
  - Syntax
    - ALTER TABLE table_name RENAME COLUMN old_name to new_name;

- Modify an existing column:
  - ALTER TABLE Employees MODIFY salary DECIMAL(10, 2) NULL;
  - ALTER TABLE Employees MODIFY salary DECIMAL(10, 2) NOT NULL;

DROP TABLE Command
DDL

- The DROP TABLE statement is used to drop an existing table in a database.
- Syntax:
  - DROP TABLE table_name;

```
 MySQL  localhost:3306 ssl  miu_db  SQL  DROP TABLE Employees;
Query OK, 0 rows affected (0.0202 sec)
 MySQL  localhost:3306 ssl  miu_db  SQL  SHOW TABLES;
Empty set (0.0023 sec)
 MySQL  localhost:3306 ssl  miu_db  SQL
```

DDL Commands
CREATE
DROP
ALTER
TRUNCATE
RENAME

TRUNCATE TABLE Command
DDL

- The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.
- Faster than DELETE because it does not log individual row deletions.
- Syntax
  - TRUNCATE TABLE table_name;

```
MySQL  localhost:3306 ssl  miu_db  SQL  TRUNCATE TABLE Employees;

                    OR

MySQL  localhost:3306 ssl  miu_db  SQL  TRUNCATE Employees;
```

DDL Commands
CREATE
DROP
ALTER
TRUNCATE
RENAME

RENAME TABLE Command

DDL

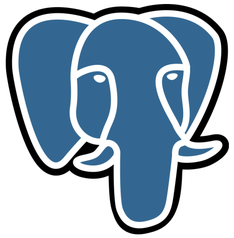
DDL Commands
CREATE
DROP
ALTER
TRUNCATE
RENAME

- To rename a column in a table, use the following syntax:
- Syntax
  - RENAME TABLE table_name TO new_table_name;
  - ALTER TABLE table_name RENAME [TO|AS] new_table_name;

RENAME TABLE Command

```
MySQL localhost:3306 ssl miu_db SQL > ALTER TABLE Employees RENAME TO Staff;
Query OK, 0 rows affected (0.0100 sec)
MySQL localhost:3306 ssl miu_db SQL > SHOW TABLES;
+-----------------+
| Tables_in_miu_db |
+-----------------+
| Staff           |
+-----------------+
1 row in set (0.0037 sec)
MySQL localhost:3306 ssl miu_db SQL > ALTER TABLE Staff RENAME as Employees;
Query OK, 0 rows affected (0.0099 sec)
MySQL localhost:3306 ssl miu_db SQL > SHOW TABLES;
+-----------------+
| Tables_in_miu_db |
+-----------------+
| Employees       |
+-----------------+
1 row in set (0.0010 sec)
MySQL localhost:3306 ssl miu_db SQL > RENAME TABLE Employees TO Staff;
Query OK, 0 rows affected (0.0106 sec)
MySQL localhost:3306 ssl miu_db SQL > SHOW TABLES;
+-----------------+
| Tables_in_miu_db |
+-----------------+
| Staff           |
+-----------------+
1 row in set (0.0010 sec)
MySQL localhost:3306 ssl miu_db SQL >
```

DDL Commands
CREATE
DROP
ALTER
TRUNCATE
RENAME



RENAME TABLE Command

Postgresql

```
smartcitydb=# \dt librarydb.*
 librarydb | employees | table | bright

smartcitydb=# ALTER TABLE librarydb.employees RENAME TO Staff;
ALTER TABLE
smartcitydb=# \dt librarydb.*
 librarydb | staff | table | bright
```

DDL Commands
CREATE
DROP
ALTER
TRUNCATE
RENAME

Note:

Postgresql does not support the following command:

```
RENAME TABLE employees TO Staff;
```

Outline

- Introduction to SQL commands
- DDL
- DQL
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

Data Manipulation Language
DML

- It provides a set of operations to support the basic data manipulation operations on the data held in database.
- Data manipulation operations usually include:


Insertion of new data into the database


Modification of data stored in the database


Deletion of data from the database

Data Manipulation Language

DML

INSERT – to insert data into a table

UPDATE – to update data in a table

DELETE – to delete data from a table

Data Query Language

DQL

Data Query Language (DQL) is a component of Structured Query Language (SQL) focused on retrieving data from databases using the SELECT statement.

SELECT Statement

- The purpose of the SELECT statement is to retrieve and display data from one or more database tables.
- It is an extremely powerful command, capable of performing the equivalent of the relational algebra's Selection, Projection, and Join operations in a single statement.

DML Commands
SELECT
INSERT
UPDATE
DELETE

SELECT Statement

General Form

SELECT [DISTINCT | ALL] {* | [columnExpression [AS newName]] [, . . .]}

FROM TableName [alias] [, . . .]

[WHERE condition]

[GROUP BY columnList] [HAVING condition]

[ORDER BY columnList]

FROM specifies the table or tables to be used

WHERE filters the rows subject to some condition

GROUP BY forms groups of rows with the same column value

HAVING filters the groups subject to some condition

SELECT specifies which columns are to appear in the output

ORDER BY specifies the order of the output

SELECT Statement

1. Retrieve all columns, all rows

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

```
MySQL localhost:3306 ssl miu_db SQL SELECT staffNo, fName, lName, position, sex, DOB, salary, branchNo FROM Staff;
+---------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+---------+-------+-------+------------+-----+------------+----------+----------+
| SL21    | John  | White | Manager    | M   | 1945-10-01 | 30000.00 | B005     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SA9     | Mary  | Howe  | Assistant  | F   | 1970-02-19 | 90000.00 | B007     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 24000.00 | B003     |
| SL41    | Julie | Lee   | Assistant  | F   | 1965-06-13 |  9000.00 | B005     |
+---------+-------+-------+------------+-----+------------+----------+----------+
6 rows in set (0.0025 sec)
MySQL localhost:3306 ssl miu_db SQL
```

Use asterik (*) to represent all columns

```
MySQL localhost:3306 ssl miu_db SQL SELECT * FROM Staff;
```

DATE_FORMAT() Function

MySQL only

Format	Description
%a	Abbreviated weekday name (Sun to Sat)
%b	Abbreviated month name (Jan to Dec)
%c	Numeric month name (0 to 12)
%D	Day of the month as a numeric value, followed by suffix (1st, 2nd, 3rd, ...)
%d	Day of the month as a numeric value (01 to 31)
%e	Day of the month as a numeric value (0 to 31)
%f	Microseconds (000000 to 999999)
%H	Hour (00 to 23)
%h	Hour (00 to 12)
%I	Hour (00 to 12)
%i	Minutes (00 to 59)
%j	Day of the year (001 to 366)
%k	Hour (0 to 23)
%l	Hour (1 to 12)

Format	Description
%M	Month name in full (January to December)
%m	Month name as a numeric value (00 to 12)
%p	AM or PM
%r	Time in 12 hour AM or PM format (hh:mm:ss AM/PM)
%S	Seconds (00 to 59)
%s	Seconds (00 to 59)
%T	Time in 24 hour format (hh:mm:ss)
%U	Week where Sunday is the first day of the week (00 to 53)
%u	Week where Monday is the first day of the week (00 to 53)
%V	Week where Sunday is the first day of the week (01 to 53). Used with %X
%v	Week where Monday is the first day of the week (01 to 53). Used with %x
%W	Weekday name in full (Sunday to Saturday)
%w	Day of the week where Sunday=0 and Saturday=6
%X	Year for the week where Sunday is the first day of the week. Used with %V
%x	Year for the week where Monday is the first day of the week. Used with %v
%Y	Year as a numeric, 4-digit value
%y	Year as a numeric, 2-digit value

SELECT Statement

Retrieve all columns, all rows

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

%b	Abbreviated month name (Jan to Dec)
%e	Day of the month as a numeric value (0 to 31)
%Y	Year as a numeric, 4-digit value
%y	Year as a numeric, 2-digit value

```
MySQL localhost:3306 ssl miu_db SQL > SELECT fName, lName, staffNo, position, sex, DATE_FORMAT(DOB, '%e-%b-%Y'),
salary, branchNo FROM Staff;
+-------+-------+---------+------------+-----+---------------------------------+----------+----------+
| fName | lName | staffNo | position   | sex | DATE_FORMAT(DOB, '%e-%b-%Y')    | salary   | branchNo |
+-------+-------+---------+------------+-----+---------------------------------+----------+----------+
| John  | White | SL21    | Manager    | M   | 1-Oct-1945                      | 30000.00 | B005     |
| Ann   | Beech | SG37    | Assistant  | F   | 10-Nov-1960                     | 12000.00 | B003     |
| David | Ford  | SG14    | Supervisor | M   | 24-Mar-1958                     | 18000.00 | B003     |
| Mary  | Howe  | SA9     | Assistant  | F   | 19-Feb-1970                     | 90000.00 | B007     |
| Susan | Brand | SG5     | Manager    | F   | 3-Jun-1940                      | 24000.00 | B003     |
| Julie | Lee   | SL41    | Assistant  | F   | 13-Jun-1965                     |  9000.00 | B005     |
+-------+-------+---------+------------+-----+---------------------------------+----------+----------+
6 rows in set (0.0013 sec)
MySQL localhost:3306 ssl miu_db SQL >
```

SELECT Statement

Retrieve all columns, all rows

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

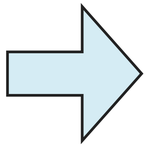
```
SELECT fName, lName, staffNo, position, sex, DATE_FORMAT(DOB, '%e-%b-%Y') AS DOB, salary, branchNo FROM Staff;
+-------+-------+---------+------------+-----+-----------------------------+----------+----------+
| fName | lName | staffNo | position   | sex | DOB                         | salary   | branchNo |
+-------+-------+---------+------------+-----+-----------------------------+----------+----------+
| John  | White | SL21    | Manager    | M   | 1-Oct-1945                  | 30000.00 | B005     |
| Ann   | Beech | SG37    | Assistant  | F   | 10-Nov-1960                 | 12000.00 | B003     |
| David | Ford  | SG14    | Supervisor | M   | 24-Mar-1958                 | 18000.00 | B003     |
| Mary  | Howe  | SA9     | Assistant  | F   | 19-Feb-1970                 | 90000.00 | B007     |
| Susan | Brand | SG5     | Manager    | F   | 3-Jun-1940                  | 24000.00 | B003     |
| Julie | Lee   | SL41    | Assistant  | F   | 13-Jun-1965                 |  9000.00 | B005     |
+---+-------+---------+------------+-----+-----------------------------+----------+----------+
6 rows in set (0.0013 sec)
 MySQL  localhost:3306 ssl  miu_db  SQL
```

SELECT Statement

2. Retrieve specific columns, all rows

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005



```
+---------+-------+-------+----------+
| staffNo | fName | lName | salary   |
+---------+-------+-------+----------+
| SL21    | John  | White | 30000.00 |
| SG37    | Ann   | Beech | 12000.00 |
| SG14    | David | Ford  | 18000.00 |
| SA9     | Mary  | Howe  | 90000.00 |
| SG5     | Susan | Brand | 24000.00 |
| SL41    | Julie | Lee   |  9000.00 |
+---------+-------+-------+----------+
```

```
MySQL localhost:3306 ssl miu_db SQL SELECT staffNo, fName, lName, salary FROM Staff;
+---------+-------+-------+----------+
| staffNo | fName | lName | salary   |
+---------+-------+-------+----------+
| SL21    | John  | White | 30000.00 |
| SG37    | Ann   | Beech | 12000.00 |
| SG14    | David | Ford  | 18000.00 |
| SA9     | Mary  | Howe  | 90000.00 |
| SG5     | Susan | Brand | 24000.00 |
| SL41    | Julie | Lee   |  9000.00 |
+---------+-------+-------+----------+
6 rows in set (0.0025 sec)
MySQL localhost:3306 ssl miu_db SQL
```

SELECT Statement

3. Use of DISTINCT

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	2013-05-24	too small
CR56	PG36	2013-04-28	NULL
CR56	PG4	2013-05-26	NULL
CR62	PA14	2013-05-14	no dining room
CR76	PG4	2013-04-20	too remote

```
SELECT propertyNo FROM Viewing;
+------------+
| propertyNo |
+------------+
| PA14       |
| PG36       |
| PG4        |
| PA14       |
| PG4        |
+------------+
5 rows in set (0.0016 sec)
```

MySQL localhost:3306 ssl miu_db SQL

To eliminate the duplicates, we use the DISTINCT keyword

```
SELECT DISTINCT propertyNo FROM Viewing;
+------------+
| propertyNo |
+------------+
| PA14       |
| PG36       |
| PG4        |
+------------+
3 rows in set (0.0034 sec)
```

SELECT Statement

3. Use of DISTINCT

Orders

*	customerid	customername	address	city	phone	email	product
1	1	John Smith	123 Main St, Apt 4A	New York	(555) 555-1234	john.smith@email.com	Smartphone
2	2	Jane Doe	456 Elm St, Unit 7	Los Angeles	(555) 555-5678	jane.doe@email.com	Laptop
3	3	Robert Brown	789 Oak Dr, #22	Chicago	(555) 555-9876	robert.brown@email.com	Tablet
4	4	Susan Lee	101 Pine Rd, #5B	New York	(555) 555-2468	susan.lee@email.com	Smartwatch
5	5	Michael White	321 Cedar St, #9	San Francisco	(555) 555-4321	michael.white@email.com	Laptop
6	6	Sarah Johnson	222 Elm St, Apt 12	Los Angeles	(555) 555-9999	sarah.johnson@email.com	Smartphone
7	7	David Davis	444 Oak St, #18	Chicago	(555) 555-1111	david.davis@email.com	Smart TV
8	8	Emily Wilson	555 Pine Rd, #3C	New York	(555) 555-8888	emily.wilson@email.com	Headphones
9	9	Richard Harris	777 Cedar St, #5A	San Francisco	(555) 555-7777	richard.harris@email.com	Tablet
10	10	Maria Lopez	888 Elm St, Unit 15	Los Angeles	(555) 555-3333	maria.lopez@email.com	Smartphone
11	11	William Clark	999 Pine Rd, #8	Chicago	(555) 555-5555	william.clark@email.com	Smartwatch
12	12	Catherine Hall	111 Oak Dr, #14	New York	(555) 555-6666	catherine.hall@email.com	Tablet
13	13	Daniel Martinez	333 Cedar St, #6B	San Francisco	(555) 555-2222	daniel.martinez@email.com	Headphones
14	14	Laura Taylor	444 Main St, Apt 3D	Los Angeles	(555) 555-4444	laura.taylor@email.com	Phone Charger
15	15	George Anderson	555 Elm St, Unit 11	Chicago	(555) 555-0000	george.anderson@email.co...	Microphone
16	16	Linda Rodriguez	666 Oak Dr, #12	New York	(555) 555-9999	linda.rodriguez@email.com	Smartphone
17	17	Matthew Brown	777 Pine Rd, #7A	San Francisco	(555) 555-1111	matthew.brown@email.com	Laptop
18	18	Jennifer Allen	888 Cedar St, #8B	Los Angeles	(555) 555-3333	jennifer.allen@email.com	Smart TV
19	19	James Baker	999 Elm St, Unit 9	Chicago	(555) 555-5555	james.baker@email.com	Microphone
20	20	Patricia Davis	111 Pine Rd, #4C	New York	(555) 555-6666	patricia.davis@email.com	Headphones

Get a list of unique cities where the customers are located

```
SELECT DISTINCT city FROM orders;
```

city
New York
San Fransisco
Chicago
Los Angeles

Ref:
https://www.dbvis.com/thetable/sql-distinct-a-comprehensive-guide/

SELECT Statement

3. Use of DISTINCT

E-Mart wants to thank each customer for shopping in different cities. The marketing team needs a list of every unique combination of customer name, customer email, and city—with each combination appearing only once, even if the same customer placed multiple orders from the same city.

```
+----------+--------------+-----------------+---------+---------+
| order_id | customerName | email           | city    | product |
+----------+--------------+-----------------+---------+---------+
|        1 | Alice        | alice@mail.com | Chicago | Book    |
|        2 | Alice        | alice@mail.com | Chicago | Pen     |
|        3 | Alice        | alice@mail.com | Chicago | Book    |
|        4 | Bob          | bob@mail.com   | Boston  | Book    |
+----------+--------------+-----------------+---------+---------+
```

```
MySQL localhost:3306 ssl ecommerce_db SQL
SELECT DISTINCT customerName, email, city
FROM Orders;
+--------------+----------------+---------+
| customerName | email          | city    |
+--------------+----------------+---------+
| Alice        | alice@mail.com | Chicago |
| Bob          | bob@mail.com   | Boston  |
+--------------+----------------+---------+
```

SELECT Statement

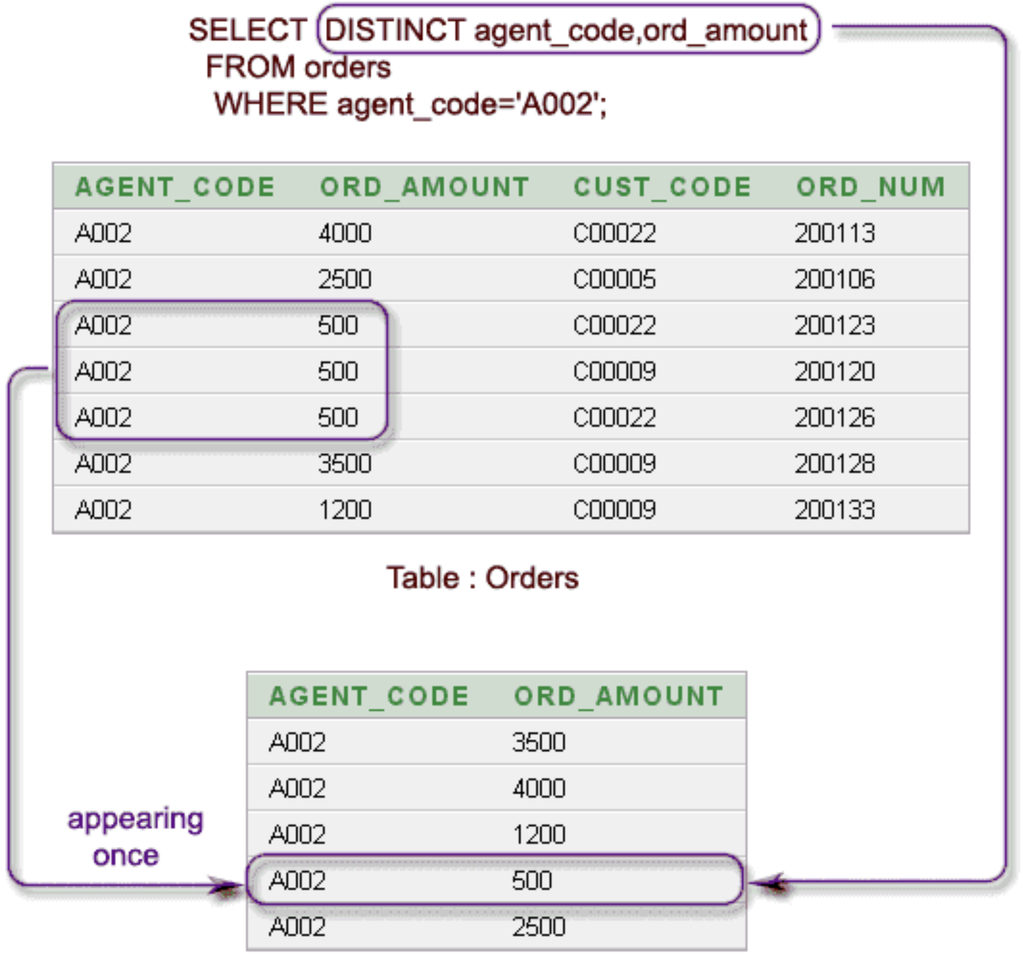
3. Use of DISTINCT

The management team wants to know how many unique customers have placed orders in each city.

```
+----------+--------------+-----------------+---------+---------+
| order_id | customerName | email           | city    | product |
+----------+--------------+-----------------+---------+---------+
|        1 | Alice        | alice@mail.com | Chicago | Book    |
|        2 | Alice        | alice@mail.com | Chicago | Pen     |
|        3 | Alice        | alice@mail.com | Chicago | Book    |
|        4 | Bob          | bob@mail.com   | Boston  | Book    |
+----------+--------------+-----------------+---------+---------+
```

```
MySQL localhost:3306 ssl ecommerce_db SQL
SELECT city, COUNT(DISTINCT customername) AS unique_customer_count
FROM orders
GROUP BY city;
```

```
+---------+-----------------------+
| city    | unique_customer_count |
+---------+-----------------------+
| Boston  |                     1 |
| Chicago |                     1 |
+---------+-----------------------+
```



Ref:

https://www.w3resource.com/sql/select-statement/queries-with-distinct-multiple-columns.php#google_vignette

SELECT Statement

4. Calculated Fields

Produce a list of monthly salaries for all staff, showing the staff number, the first and last names, and the salary details.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

```
SELECT staffNo, fName, lName, salary/12 FROM staff;
+---------+-------+-------+-------------+
| staffNo | fName | lName | salary/12   |
+---------+-------+-------+-------------+
| SL21    | John  | White | 2500.000000 |
| SG37    | Ann   | Beech | 1000.000000 |
| SG14    | David | Ford  | 1500.000000 |
| SA9     | Mary  | Howe  | 7500.000000 |
| SG5     | Susan | Brand | 2000.000000 |
| SL41    | Julie | Lee   |  750.000000 |
+---------+-------+-------+-------------+
6 rows in set (0.0040 sec)
MySQL  localhost:3306 ssl  miu_db  SQL
```

(sometimes called a computed or derived field)

An SQL expression can involve addition, subtraction, multiplication, and division, and parentheses can be used to build complex expressions. More than one table column can be used in a calculated column; however, the columns referenced in an arithmetic expression must have a numeric type.

SELECT Statement

Calculated Fields

Produce a list of monthly salaries for all staff, showing the staff number, the first and last names, and the salary details.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

```
SELECT staffNo, fName, lName, salary/12 AS monthlySalary FROM staff;
+---------+-------+-------+---------------+
| staffNo | fName | lName | monthlySalary |
+---------+-------+-------+---------------+
| SL21    | John  | White |   2500.000000 |
| SG37    | Ann   | Beech |   1000.000000 |
| SG14    | David | Ford  |   1500.000000 |
| SA9     | Mary  | Howe  |   7500.000000 |
| SG5     | Susan | Brand |   2000.000000 |
| SL41    | Julie | Lee   |    750.000000 |
+---------+-------+-------+---------------+
6 rows in set (0.0014 sec)
 MySQL  localhost:3306 ssl  miu_db  SQL
```

SELECT Statement - Row Selection
WHERE clause

- Restrict the rows that are retrieved.
- This can be achieved with the WHERE clause, which consists of the keyword **WHERE followed by a search condition** that specifies the rows to be retrieved.
- The five basic search conditions (or predicates, using the ISO terminology) are as follows:

• *Comparison*	Compare the value of one expression to the value of another expression.
• *Range*	Test whether the value of an expression falls within a specified range of values.
• *Set membership*	Test whether the value of an expression equals one of a set of values.
• *Pattern match*	Test whether a string matches a specified pattern.
• *Null*	Test whether a column has a null (unknown) value.

SELECT Statement - Row Selection

5. Comparison Search Condition

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

Syntax:

SELECT column_names

FROM table_name

WHERE <expression> [comparison operator] <expression>;

SELECT Statement - Row Selection

Comparison Search Condition

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

- List all staff with a salary greater than $10,000.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

SELECT column_names

FROM table_name

WHERE <expression> [comparison operator] <expression>;

```
MySQL  localhost:3306 ssl  miu_db  SQL
SELECT *
FROM Staff
WHERE salary > 10000;
+---------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+---------+-------+-------+------------+-----+------------+----------+----------+
| SL21    | John  | White | Manager    | M   | 1945-10-01 | 30000.00 | B005     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SA9     | Mary  | Howe  | Assistant  | F   | 1970-02-19 | 90000.00 | B007     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 24000.00 | B003     |
+---------+-------+-------+------------+-----+------------+----------+----------+
```

SELECT Statement - Row Selection

Comparison Search Condition

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

Category	Operator	Description	Evaluation Order
Comparison Operator	=	Equals	–
	<>	Is not equal to (ISO standard)	–
	!=	Is not equal to (some dialects)	–
	<	Is less than	–
	<=	Is less than or equal to	–
	>	Is greater than	–
	>=	Is greater than or equal to	–
Logical Operator	NOT	Negates a condition	1
	AND	Both conditions must be true	2
	OR	At least one condition must be true	3
Evaluation Rule	()	Parentheses: expressions within are evaluated first	0 (highest precedence)
Evaluation Direction	–	Left to right within the same precedence level	Applies to all levels

SELECT Statement - Row Selection

6. Compound Comparison Search Condition

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

Syntax:

SELECT column_names

FROM table_name

WHERE condition1 AND condition2 AND ...conditionN;

SELECT Statement - Row Selection

6.Compound Comparison Search Condition

- Comparison
- Range
- Set membership
- Pattern match
- Null

- List all staff whose positions are either Manager or Assistant.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

SELECT column_names
FROM table_name
WHERE condition1 AND condition2 AND ...conditionN;

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Staff
                                        WHERE position = 'Manager' OR position = 'Assistant';
+---------+-------+-------+-----------+-----+------------+----------+----------+
| staffNo | fName | lName | position  | sex | DOB        | salary   | branchNo |
+---------+-------+-------+-----------+-----+------------+----------+----------+
| SL21    | John  | White | Manager   | M   | 1945-10-01 | 30000.00 | B005     |
| SG37    | Ann   | Beech | Assistant | F   | 1960-11-10 | 12000.00 | B003     |
| SA9     | Mary  | Howe  | Assistant | F   | 1970-02-19 | 90000.00 | B007     |
| SG5     | Susan | Brand | Manager   | F   | 1940-06-03 | 24000.00 | B003     |
| SL41    | Julie | Lee   | Assistant | F   | 1965-06-13 |  9000.00 | B005     |
+---------+-------+-------+-----------+-----+------------+----------+----------+
5 rows in set (0.0023 sec)
```

SELECT Statement - Row Selection

6.Compound Comparison Search Condition

- List all staff whose positions are either Manager or Assistant.

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

```
SELECT staffNo, fName, lName, position FROM Staff WHERE position IN ('manager', 'supervisor');
SELECT staffNo, fName, lName, position FROM Staff WHERE position ='manager' OR position= 'supervisor';
```

SELECT Statement

7. Range Search Condition

Syntax:

```
SELECT column_names
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Note:

The BETWEEN test includes the endpoints of the range.

SELECT Statement - Row Selection

7. Range Search Condition

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

- List all staff with a salary between $ 20,000 and $ 30,000.

Staff

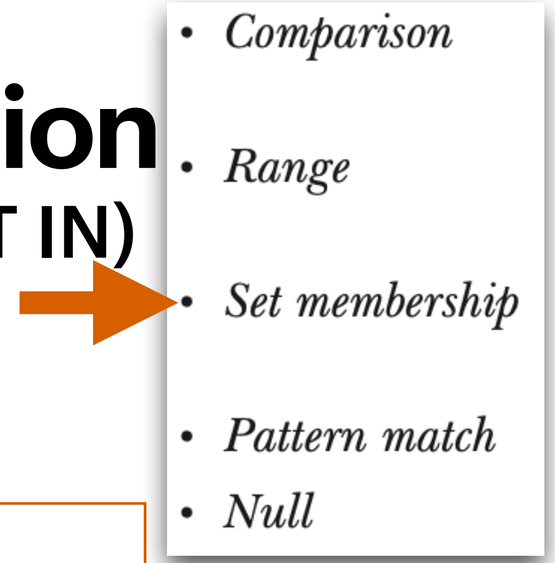
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

```
SELECT column_names
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT staffNo, fName, lName, position, salary
                                        FROM Staff
                                        WHERE salary BETWEEN 20000 AND 30000;
+---------+-------+-------+----------+----------+
| staffNo | fName | lName | position | salary   |
+---------+-------+-------+----------+----------+
| SL21    | John  | White | Manager  | 30000.00 |
| SG5     | Susan | Brand | Manager  | 24000.00 |
+---------+-------+-------+----------+----------+
2 rows in set (0.0024 sec)
```

SELECT Statement - Row Selection

8. Set Membership Search Condition (IN / NOT IN)



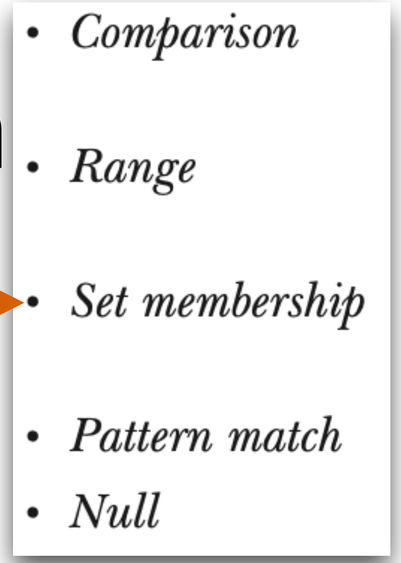
Syntax:

```
SELECT column_names
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

```
SELECT column_names
FROM table_name
WHERE column_name NOT IN (value1, value2, ...);
```

SELECT Statement - Row Selection

8. Set Membership Search Condition (IN / NOT IN)



- List all managers and supervisors

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

```
SELECT column_names
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT staffNo, fName, lName, position
                                        FROM Staff
                                        WHERE position IN ('manager', 'supervisor');
+---------+-------+-------+------------+
| staffNo | fName | lName | position   |
+---------+-------+-------+------------+
| SL21    | John  | White | Manager    |
| SG14    | David | Ford  | Supervisor |
| SG5     | Susan | Brand | Manager    |
+---------+-------+-------+------------+
3 rows in set (0.0005 sec)
```

SELECT Statement - Row Selection

9. Pattern Match Search Condition (LIKE / NOT LIKE)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

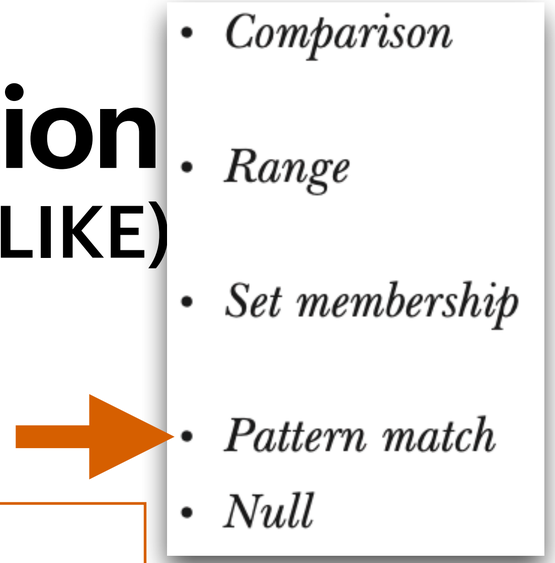
Syntax:

```
SELECT column_names
FROM table_name
WHERE column LIKE pattern;
```

```
SELECT column_names
FROM table_name
WHERE column NOT LIKE pattern;
```

SELECT Statement - Row Selection

9. Pattern Match Search Condition (LIKE / NOT LIKE)



SQL has two special pattern-matching symbols:

- The % percent character represents any sequence of zero or more characters (wildcard).
- The _ underscore character represents any single character.

SELECT Statement - Row Selection

9. Pattern Match Search Condition (LIKE / NOT LIKE)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

```
SELECT column_names
FROM table_name
WHERE column LIKE pattern;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

fName LIKE ‘J%’ means the first character must be J, but the rest of the string can be anything.

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Staff
                                        WHERE fName LIKE 'J%';
+---------+-------+-------+-----------+-----+------------+----------+----------+
| staffNo | fName | lName | position  | sex | DOB        | salary   | branchNo |
+---------+-------+-------+-----------+-----+------------+----------+----------+
| SL21    | John  | White | Manager   | M   | 1945-10-01 | 30000.00 | B005     |
| SL41    | Julie | Lee   | Assistant | F   | 1965-06-13 |  9000.00 | B005     |
+---------+-------+-------+-----------+-----+------------+----------+----------+
2 rows in set (0.0004 sec)
MySQL  localhost:3306 ssl  miu_db  SQL
```

SELECT Statement - Row Selection

9. Pattern Match Search Condition (LIKE / NOT LIKE)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

```
SELECT column_names
FROM table_name
WHERE column LIKE pattern;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

fName LIKE ‘J_ _ _ _’ means that there must be exactly five characters in the string, the first of which must be an J.

```
MySQL  localhost:3306 ssl  miu_db  SQL
SELECT *
FROM Staff
WHERE fName LIKE 'J____';
+---------+-------+-------+-----------+-----+------------+---------+----------+
| staffNo | fName | lName | position  | sex | DOB        | salary  | branchNo |
+---------+-------+-------+-----------+-----+------------+---------+----------+
| SL41    | Julie | Lee   | Assistant | F   | 1965-06-13 | 9000.00 | B005     |
+---------+-------+-------+-----------+-----+------------+---------+----------+
1 row in set (0.0005 sec)
MySQL  localhost:3306 ssl  miu_db  SQL
```

SELECT Statement - Row Selection

9. Pattern Match Search Condition (LIKE / NOT LIKE)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

```
SELECT column_names
FROM table_name
WHERE column NOT LIKE pattern;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT * FROM Staff WHERE fName NOT LIKE 'J____';
+---------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+---------+-------+-------+------------+-----+------------+----------+----------+
| SL21    | John  | White | Manager    | M   | 1945-10-01 | 30000.00 | B005     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SA9     | Mary  | Howe  | Assistant  | F   | 1970-02-19 | 90000.00 | B007     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 24000.00 | B003     |
+---------+-------+-------+------------+-----+------------+----------+----------+
5 rows in set (0.0020 sec)
MySQL  localhost:3306 ssl  miu_db  SQL
```

SELECT Statement - Row Selection

9. Pattern Match Search Condition (LIKE / NOT LIKE)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

```
SELECT column_names
FROM table_name
WHERE column LIKE pattern;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

fName LIKE ‘%e’ means any sequence of characters, of length at least 1, with the last character an e.

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Staff
                                        WHERE fName LIKE '%e';
+---------+-------+-------+-----------+-----+------------+---------+----------+
| staffNo | fName | lName | position  | sex | DOB        | salary  | branchNo |
+---------+-------+-------+-----------+-----+------------+---------+----------+
| SL41    | Julie | Lee   | Assistant | F   | 1965-06-13 | 9000.00 | B005     |
+---------+-------+-------+-----------+-----+------------+---------+----------+
1 row in set (0.0005 sec)
 MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *  FROM Staff WHERE fName LIKE '%e';
```

SELECT Statement - Row Selection

9. Pattern Match Search Condition (LIKE / NOT LIKE)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

```
SELECT column_names
FROM table_name
WHERE column LIKE pattern;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

lName LIKE '%ee%' means a sequence of characters of any length containing *ee*

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Staff
                                        WHERE lName LIKE '%ee%';
+---------+-------+-------+-----------+-----+------------+----------+----------+
| staffNo | fName | lName | position  | sex | DOB        | salary   | branchNo |
+---------+-------+-------+-----------+-----+------------+----------+----------+
| SG37    | Ann   | Beech | Assistant | F   | 1960-11-10 | 12000.00 | B003     |
| SL41    | Julie | Lee   | Assistant | F   | 1965-06-13 |  9000.00 | B005     |
+---------+-------+-------+-----------+-----+------------+----------+----------+
2 rows in set (0.0008 sec)
 MySQL  localhost:3306 ssl  miu_db  SQL
```

SELECT Statement - Row Selection

10. NULL Search Condition (IS NULL / IS NOT NULL)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

Let's create a table with NULL values.

```
MySQL  localhost:3306 ssl  miu_db  SQL  CREATE TABLE Viewing (
                                            clientNo VARCHAR(4) NOT NULL,
                                            propertyNo VARCHAR(4) NOT NULL,
                                            viewDate DATE NOT NULL,
                                            comment VARCHAR(255),
                                            PRIMARY KEY (clientNo, propertyNo)
                                        );
Query OK, 0 rows affected (0.0054 sec)
MySQL  localhost:3306 ssl  miu_db  SQL  INSERT INTO Viewing (clientNo, propertyNo, viewDate, comment) VALUES
                                        ('CR56', 'PA14', '2013-05-24', 'too small'),
                                        ('CR76', 'PG4', '2013-04-20', 'too remote'),
                                        ('CR56', 'PG4', '2013-05-26', NULL),
                                        ('CR62', 'PA14', '2013-05-14', 'no dining room'),
                                        ('CR56', 'PG36', '2013-04-28', NULL);
Query OK, 5 rows affected (0.0017 sec)

Records: 5  Duplicates: 0  Warnings: 0
MySQL  localhost:3306 ssl  miu_db  SQL
```

SELECT Statement - Row Selection

10. NULL Search Condition (IS NULL / IS NOT NULL)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

List the details of all viewings on property PG4 where a comment has not been supplied.

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Viewing;
+----------+------------+------------+----------------+
| clientNo | propertyNo | viewDate   | comment        |
+----------+------------+------------+----------------+
| CR56     | PA14       | 2013-05-24 | too small      |
| CR56     | PG36       | 2013-04-28 | NULL           |
| CR56     | PG4        | 2013-05-26 | NULL           |
| CR62     | PA14       | 2013-05-14 | no dining room |
| CR76     | PG4        | 2013-04-20 | too remote     |
+----------+------------+------------+----------------+
5 rows in set (0.0004 sec)
```

Syntax:

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

SELECT Statement - Row Selection

10. NULL Search Condition (IS NULL / IS NOT NULL)

- *Comparison*
- *Range*
- *Set membership*
- *Pattern match*
- *Null*

Suppose you want to analyze only those property viewings where potential clients provided feedback for property PG4. Write a query to list all such feedbacks.

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Viewing;
+----------+------------+------------+----------------+
| clientNo | propertyNo | viewDate   | comment        |
+----------+------------+------------+----------------+
| CR56     | PA14       | 2013-05-24 | too small      |
| CR56     | PG36       | 2013-04-28 | NULL           |
| CR56     | PG4        | 2013-05-26 | NULL           |
| CR62     | PA14       | 2013-05-14 | no dining room |
| CR76     | PG4        | 2013-04-20 | too remote     |
+----------+------------+------------+----------------+
5 rows in set (0.0004 sec)
```

Solution:

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Viewing
                                        WHERE propertyNo = 'PG4' AND comment IS NOT NULL;
+----------+------------+------------+------------+
| clientNo | propertyNo | viewDate   | comment    |
+----------+------------+------------+------------+
| CR76     | PG4        | 2013-04-20 | too remote |
+----------+------------+------------+------------+
1 row in set (0.0005 sec)
MySQL  localhost:3306 ssl  miu_db  SQL
```

Sorting Results
ORDER BY Clause

- In general, the rows of an SQL query result table are not arranged in any particular order (although some DBMSs may use a default ordering based, for example, on a primary key).
- However, we can ensure the results of a query are sorted using the ORDER BY clause in the SELECT statement.
- The ORDER BY clause consists of a list of column identifiers that the result is to be sorted on, separated by commas.

Sorting Results

1. ORDER BY Clause - Single Column Ordering

- Produce a list of salaries for all staff, arranged in descending order of salary.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Syntax:

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

Sorting Results

1. ORDER BY Clause - Single Column Ordering

- Produce a list of salaries for all staff, arranged in descending order of salary.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT *
                                        FROM Staff
                                        ORDER BY salary DESC;
+---------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+---------+-------+-------+------------+-----+------------+----------+----------+
| SA9     | Mary  | Howe  | Assistant  | F   | 1970-02-19 | 90000.00 | B007     |
| SL21    | John  | White | Manager    | M   | 1945-10-01 | 30000.00 | B005     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 24000.00 | B003     |
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SL41    | Julie | Lee   | Assistant  | F   | 1965-06-13 |  9000.00 | B005     |
+---------+-------+-------+------------+-----+------------+----------+----------+
6 rows in set (0.0059 sec)
```

Sorting Results

2. ORDER BY Clause - Multiple Column Ordering

```
SELECT * FROM Staff ORDER BY salary DESC, staffNo ASC;
```

- A minor sort key (also called a secondary sort key) is the column used to break ties when two or more rows have the same value in the major (primary) sort key.

- Major sort key
  - salary DESC → sorts by salary, highest first.
- Minor sort key
  - staffNo ASC → if two staff have the same salary, this decides the order by staff number.

Sorting Results

2. ORDER BY Clause - Multiple Column Ordering

Write an SQL query that retrieves all details for every staff member from the Staff table. The results should be ordered primarily by salary in descending order, so the highest earners appear first. In cases where two or more staff members have the exact same salary, their order should then be determined by their staffNo in ascending order, ensuring a consistent and unique ordering even in the presence of salary ties.

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT * FROM Staff;
+---------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+---------+-------+-------+------------+-----+------------+----------+----------+
| SA9     | Mary  | Howe  | Assistant  | F   | 1970-02-19 | 90000.00 | B007     |
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 30000.00 | B003     |
| SL21    | John  | White | Manager    | M   | 1945-10-01 | 30000.00 | B005     |
| SL41    | Julie | Lee   | Assistant  | F   | 1965-06-13 |  9000.00 | B005     |
+---------+-------+-------+------------+-----+------------+----------+----------+
6 rows in set (0.0021 sec)
```

Solution

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT * FROM Staff ORDER BY salary DESC, staffNo ASC;
+---------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+---------+-------+-------+------------+-----+------------+----------+----------+
| SA9     | Mary  | Howe  | Assistant  | F   | 1970-02-19 | 90000.00 | B007     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 30000.00 | B003     |
| SL21    | John  | White | Manager    | M   | 1945-10-01 | 30000.00 | B005     |
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SL41    | Julie | Lee   | Assistant  | F   | 1965-06-13 |  9000.00 | B005     |
+---------+-------+-------+------------+-----+------------+----------+----------+
```

Outline

- Introduction to SQL commands
- DDL
- DML
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

SELECT Statement - Aggregate

- To perform some form of summation or aggregation of data, similar to the totals at the bottom of a report.
- The ISO standard defines five aggregate functions:
  - COUNT – returns the number of values in a specified column
  - SUM – returns the sum of the values in a specified column
  - AVG – returns the average of the values in a specified column
  - MIN – returns the smallest value in a specified column
  - MAX – returns the largest value in a specified column

SELECT Statement - Aggregate

Create a table: PropertForRent

```
MySQL  localhost:3306 ssl  property_db  SQL  CREATE TABLE PropertyForRent (
                                                 propertyNo VARCHAR(5) PRIMARY KEY,
                                                 street VARCHAR(50) NOT NULL,
                                                 city VARCHAR(30) NOT NULL,
                                                 postcode VARCHAR(10) NOT NULL,
                                                 type VARCHAR(10) NOT NULL,
                                                 rooms INT NOT NULL,
                                                 rent INT NOT NULL,
                                                 ownerNo VARCHAR(5) NOT NULL,
                                                 staffNo VARCHAR(5), -- This column can be NULL as per your data
                                                 branchNo VARCHAR(5) NOT NULL
                                             );
Query OK, 0 rows affected (0.0064 sec)
```

```
MySQL  localhost:3306 ssl  property_db  SQL  INSERT INTO PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo) VALUES
                                             ('PA14', '16 Holhead', 'Aberdeen', 'AB7 5SU', 'House', 6, 650, 'CO46', 'SA9', 'B007'),
                                             ('PL94', '6 Argyll St', 'London', 'NW2', 'Flat', 4, 400, 'CO87', 'SL41', 'B005'),
                                             ('PG4', '6 Lawrence St', 'Glasgow', 'G11 9QX', 'Flat', 3, 350, 'CO40', NULL, 'B003'),
                                             ('PG36', '2 Manor Rd', 'Glasgow', 'G32 4QX', 'Flat', 3, 375, 'CO93', 'SG37', 'B003'),
                                             ('PG21', '18 Dale Rd', 'Glasgow', 'G12', 'House', 5, 600, 'CO87', 'SG37', 'B003'),
                                             ('PG16', '5 Novar Dr', 'Glasgow', 'G12 9AX', 'Flat', 4, 450, 'CO93', 'SG14', 'B003');
Query OK, 6 rows affected (0.0022 sec)
```

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

SELECT Statement - Aggregate

1. Use of COUNT(*)

- count(*) is used to count the number of rows in a table

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

Syntax:

```
SELECT COUNT(column_name)

FROM table_name;
```

SELECT Statement - Aggregate

1. Use of COUNT(*)

```
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street        | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead    | Aberdeen | AB7 5SU  | House |     6 |  650 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr    | Glasgow  | G12 9AX  | Flat  |     4 |  450 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd    | Glasgow  | G12      | House |     5 |  600 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd    | Glasgow  | G32 4QX  | Flat  |     3 |  375 | CO93    | SG37    | B003     |
| PG4        | 6 Lawrence St | Glasgow  | G11 9QX  | Flat  |     3 |  350 | CO40    | NULL    | B003     |
| PL94       | 6 Argyll St   | London   | NW2      | Flat  |     4 |  400 | CO87    | SL41    | B005     |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
```

```
MySQL  localhost:3306 ssl  property_db  SQL  SELECT COUNT(*)
                                             FROM PropertyForRent;
+----------+
| COUNT(*) |
+----------+
|        6 |
+----------+
1 row in set (0.0008 sec)
MySQL  localhost:3306 ssl  property_db  SQL  SELECT COUNT(*) AS TOTAL_ROWS
                                             FROM PropertyForRent;
+------------+
| TOTAL_ROWS |
+------------+
|          6 |
+------------+
1 row in set (0.0008 sec)
MySQL  localhost:3306 ssl  property_db  SQL
```

SELECT Statement - Aggregate

1. Use of COUNT(*)

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

```
MySQL  localhost:3306 ssl  property_db  SQL  SELECT COUNT(propertyNo) AS noOfProperties
                                             FROM PropertyForRent;
+----------------+
| noOfProperties |
+----------------+
|              6 |
+----------------+
1 row in set (0.0009 sec)
```

SELECT Statement - Aggregate

1. Use of COUNT(*)

How many streets?

```
MySQL localhost:3306 ssl property_db SQL > SELECT *
                                           FROM PropertyForRent;
+------------+-------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street      | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+-------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead  | Aberdeen | AB7 5SU  | House |     6 |  650 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr  | Glasgow  | G12 9AX  | Flat  |     4 |  450 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd  | Glasgow  | G12      | House |     5 |  600 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd  | Glasgow  | G32 4QX  | Flat  |     3 |  375 | CO93    | SG37    | B003     |
| PG4        | NULL        | Glasgow  | G11 9QX  | Flat  |     3 |  350 | CO40    | NULL    | B003     |
| PL94       | 6 Argyll St | London   | NW2      | Flat  |     4 |  400 | CO87    | SL41    | B005     |
+------------+-------------+----------+----------+-------+-------+------+---------+---------+----------+
6 rows in set (0.0006 sec)
MySQL localhost:3306 ssl property_db SQL > SELECT COUNT(street) AS noOfStreets
                                           FROM PropertyForRent;
+-------------+
| noOfStreets |
+-------------+
|           5 |
+-------------+
1 row in set (0.0004 sec)
MySQL localhost:3306 ssl property_db SQL >
```

If you specify a column name instead of (*), NULL values will not be counted.

SELECT Statement - Aggregate

1. Use of COUNT(*)

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

How many properties cost more than £350 per month to rent?

```
MySQL  localhost:3306 ssl  property_db  SQL
SELECT COUNT(*) AS myCount
FROM PropertyForRent
WHERE rent > 350;
+---------+
| myCount |
+---------+
|       5 |
+---------+
1 row in set (0.0026 sec)
```

SELECT Statement - Aggregate

1. Use of COUNT(*)

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

How many cities are available in this table?

```
MySQL  localhost:3306 ssl  property_db  SQL
SELECT COUNT(city) as noOfCities
FROM PropertyForRent;
+------------+
| noOfCities |
+------------+
|          6 |
+------------+
1 row in set (0.0008 sec)
```

SELECT Statement - Aggregate

2. Use of COUNT(DISTINCT)

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

How many cities are available in this table?

~~COUNT(city)~~

```
MySQL  localhost:3306 ssl  property_db  SQL
SELECT COUNT(DISTINCT city) as noOfCities
FROM PropertyForRent;
+------------+
| noOfCities |
+------------+
|          3 |
+------------+
1 row in set (0.0005 sec)
```

SELECT Statement - Aggregate

2. Use of COUNT(DISTINCT)

```
MySQL  localhost:3306 ssl  miu_db  SQL  > SELECT *
                                        FROM Viewing;
+----------+------------+------------+-----------------+
| clientNo | propertyNo | viewDate   | comment         |
+----------+------------+------------+-----------------+
| CR56     | PA14       | 2013-05-24 | too small       |
| CR56     | PG36       | 2013-04-28 | NULL            |
| CR56     | PG4        | 2013-05-26 | NULL            |
| CR62     | PA14       | 2013-05-14 | no dining room  |
| CR76     | PG4        | 2013-04-20 | too remote      |
+----------+------------+------------+-----------------+
```

How many different properties were viewed in May 2013?

```
MySQL  localhost:3306 ssl  miu_db  SQL  > SELECT COUNT(DISTINCT propertyNo) AS myCount
                                        FROM Viewing
                                        WHERE viewDate BETWEEN '2013-05-01' AND '2013-05-31';
+---------+
| myCount |
+---------+
|       2 |
+---------+
1 row in set (0.0031 sec)
```

SELECT Statement - Aggregate

3. Use of SUM

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

Syntax

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

SELECT Statement - Aggregate

3. Use of SUM

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

🤔 Find the total number of owner number and the sum of rents in the city, Glasgow.

```
MySQL  localhost:3306 ssl  property_db  SQL
SELECT COUNT(ownerNo) AS noOfOwners, SUM(rent) AS totalRent
FROM PropertyForRent
WHERE city = 'Glasgow';
+------------+-----------+
| noOfOwners | totalRent |
+------------+-----------+
|          4 |      1775 |
+------------+-----------+
1 row in set (0.0018 sec)
```

SELECT Statement - Aggregate

4. Use of MIN, Max, AVG

Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

SELECT Statement - Aggregate

4. Use of MIN, Max, AVG

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

🤔 Find the minimum, maximum, and average rent.

```
MySQL  localhost:3306 ssl  property_db  SQL
SELECT MIN(rent) AS minRent, MAX(rent) AS maxRent, AVG(rent) as avgRent
FROM PropertyForRent;
+---------+---------+----------+
| minRent | maxRent | avgRent  |
+---------+---------+----------+
|     350 |     650 | 470.8333 |
+---------+---------+----------+
1 row in set (0.0035 sec)
```

SELECT Statement - Aggregate

4. Use of MIN, Max, AVG

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

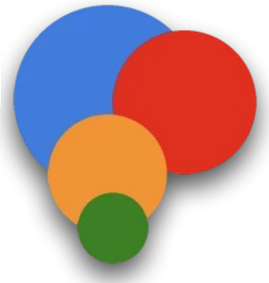
🤔 Find the minimum, maximum, and average rent where city is Glasgow.

```
MySQL localhost:3306 ssl property_db SQL
SELECT MIN(rent) AS minRent, MAX(rent) AS maxRent, AVG(rent) as avgRent
FROM PropertyForRent
WHERE city = 'Glasgow';
+---------+---------+----------+
| minRent | maxRent | avgRent  |
+---------+---------+----------+
|     350 |     600 | 443.7500 |
+---------+---------+----------+
1 row in set (0.0006 sec)
```

Outline

- Introduction to SQL commands
- DDL
- DML
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

SELECT Statement - Grouping



- To group the data from the SELECT table(s) and produces a summary row for each group.
- The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country”.
- All column names in the SELECT list must appear in the GROUP BY clause unless the name is used only in an aggregate function.
- When GROUP BY is used, the SELECT clause may only contain:
  - column names
  - aggregate functions
  - constants
  - expression involving combinations of the above
- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying predicate.
- If the GROUP BY clause is omitted when an aggregate function is used, then the entire table is considered as one group, and the aggregate function displays a single value for the entire table.

SELECT Statement - Grouping

1. Use of GROUP BY

Syntax

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

SELECT Statement - Grouping

1. Use of GROUP BY

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

```
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street        | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead    | Aberdeen | AB7 5SU  | House |     6 |  650 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr    | Glasgow  | G12 9AX  | Flat  |     4 |  450 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd    | Glasgow  | G12      | House |     5 |  600 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd    | Glasgow  | G32 4QX  | Flat  |     3 |  375 | CO93    | SG37    | B003     |
| PG4        | 6 Lawrence St | Glasgow  | G11 9QX  | Flat  |     3 |  350 | CO40    | NULL    | B003     |
| PL94       | 6 Argyll St   | London   | NW2      | Flat  |     4 |  400 | CO87    | SL41    | B005     |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
```

🤔 Count Properties by City and Type

```
MySQL localhost:3306 ssl property_db SQL
```

```
SELECT city, type, COUNT(propertyNo) AS noOfProperties
FROM PropertyForRent
GROUP BY city, type;
```

```
+----------+-------+----------------+
| city     | type  | noOfProperties |
+----------+-------+----------------+
| Aberdeen | House |              1 |
| Glasgow  | Flat  |              3 |
| Glasgow  | House |              1 |
| London   | Flat  |              1 |
+----------+-------+----------------+
4 rows in set (0.0007 sec)
```

SELECT Statement - Grouping

1. Use of GROUP BY

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

```
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street        | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead    | Aberdeen | AB7 5SU  | House |     6 |  650 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr    | Glasgow  | G12 9AX  | Flat  |     4 |  450 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd    | Glasgow  | G12      | House |     5 |  600 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd    | Glasgow  | G32 4QX  | Flat  |     3 |  375 | CO93    | SG37    | B003     |
| PG4        | 6 Lawrence St | Glasgow  | G11 9QX  | Flat  |     3 |  350 | CO40    | NULL    | B003     |
| PL94       | 6 Argyll St   | London   | NW2      | Flat  |     4 |  400 | CO87    | SL41    | B005     |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
```

🤔 Display the number of properties of type 'House' available in each city from the PropertyForRent table.

MySQL localhost:3306 ssl property_db SQL [redacted]

```
SELECT city, type, COUNT(propertyNo) AS noOfProperties
FROM PropertyForRent
WHERE type = 'House'
GROUP BY city, type;
```

```
+----------+-------+-----------------+
| city     | type  | noOfProperties  |
+----------+-------+-----------------+
| Aberdeen | House |               1 |
| Glasgow  | House |               1 |
+----------+-------+-----------------+
2 rows in set (0.0005 sec)
```

SELECT Statement - Grouping

1. Use of GROUP BY

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

```
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street        | city     | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead    | Aberdeen | AB7 5SU  | House |     6 |  650 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr    | Glasgow  | G12 9AX  | Flat  |     4 |  450 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd    | Glasgow  | G12      | House |     5 |  600 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd    | Glasgow  | G32 4QX  | Flat  |     3 |  375 | CO93    | SG37    | B003     |
| PG4        | 6 Lawrence St | Glasgow  | G11 9QX  | Flat  |     3 |  350 | CO40    | NULL    | B003     |
| PL94       | 6 Argyll St   | London   | NW2      | Flat  |     4 |  400 | CO87    | SL41    | B005     |
+------------+---------------+----------+----------+-------+-------+------+---------+---------+----------+
```

🤔 Display the number of properties of type 'House' available in each city from the PropertyForRent table.

MySQL localhost:3306 ssl property_db SQL

```
SELECT city, type, count(propertyNo) as noOfProperties
FROM PropertyForRent
WHERE type = 'House'
GROUP BY city;
```

```
+----------+-------+-----------------+
| city     | type  | noOfProperties  |
+----------+-------+-----------------+
| Aberdeen | House |               1 |
| Glasgow  | House |               1 |
+----------+-------+-----------------+
2 rows in set (0.0006 sec)
```

If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows

Restricting Groupings
HAVING clause

- The HAVING clause is designed for use with the GROUP BY clause to restrict the groups that appear in the final result table.
- Although similar in syntax, HAVING and WHERE serve different purposes. The WHERE clause filters individual rows going into the final result table, whereas HAVING filters groups going into the final result table.

Syntax

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

SELECT Statement - HAVING

Use of HAVING

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

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

Consider this query:

MySQL localhost:3306 ssl property_db SQL

```
SELECT city, type, COUNT(propertyNo) AS noOfProperties
FROM PropertyForRent
WHERE type = 'Flat'
GROUP BY city;
```

city	type	noOfProperties
Glasgow	Flat	3
London	Flat	1

🤔 What if we want only cities with more than 1 flat?

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	NULL	B003
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005

Consider this query:

MySQL localhost:3306 ssl property_db SQL

```
SELECT city, type, COUNT(propertyNo) AS noOfProperties
FROM PropertyForRent
WHERE type = 'Flat'
GROUP BY city, type;
```

city	type	noOfProperties
Glasgow	Flat	3
London	Flat	1

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

🤔 What if we want only cities with more than 1 flat?

MySQL localhost:3306 ssl property_db SQL

```
SELECT city, type, COUNT(propertyNo) AS noOfProperties
FROM PropertyForRent
WHERE type = 'Flat'
GROUP BY city, type
HAVING noOfProperties > 1;
```

city	type	noOfProperties
Glasgow	Flat	3

Note that an aggregate function can be used only in the SELECT list and in the HAVING clause.
It is incorrect to use it elsewhere.

```
+------------+---------------+---------+----------+-------+-------+------+---------+---------+----------+
| propertyNo | street        | city    | postcode | type  | rooms | rent | ownerNo | staffNo | branchNo |
+------------+---------------+---------+----------+-------+-------+------+---------+---------+----------+
| PA14       | 16 Holhead    | Aberdeen| AB7 5SU  | House |     6 |  650 | CO46    | SA9     | B007     |
| PG16       | 5 Novar Dr    | Glasgow | G12 9AX  | Flat  |     4 |  450 | CO93    | SG14    | B003     |
| PG21       | 18 Dale Rd    | Glasgow | G12      | House |     5 |  600 | CO87    | SG37    | B003     |
| PG36       | 2 Manor Rd    | Glasgow | G32 4QX  | Flat  |     3 |  375 | CO93    | SG37    | B003     |
| PG4        | 6 Lawrence St | Glasgow | G11 9QX  | Flat  |     3 |  350 | CO40    | NULL    | B003     |
| PL94       | 6 Argyll St   | London  | NW2      | Flat  |     4 |  400 | CO87    | SL41    | B005     |
+------------+---------------+---------+----------+-------+-------+------+---------+---------+----------+
```

Consider this query:

```
MySQL localhost:3306 ssl property_db SQL
SELECT city, type, COUNT(propertyNo) AS noOfProperties
FROM PropertyForRent
WHERE type = 'Flat'
GROUP BY city, type;
```

```
+---------+------+-----------------+
| city    | type | noOfProperties  |
+---------+------+-----------------+
| Glasgow | Flat |               3 |
| London  | Flat |               1 |
+---------+------+-----------------+
```

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

🤔 What if we want only cities with more than 1 flat?

```
MySQL localhost:3306 ssl property_db SQL
SELECT city, type, COUNT(propertyNo) AS noOfProperties
FROM PropertyForRent
WHERE type = 'Flat'
GROUP BY city, type
HAVING COUNT(propertyNo) > 1;
```

```
+---------+------+-----------------+
| city    | type | noOfProperties  |
+---------+------+-----------------+
| Glasgow | Flat |               3 |
+---------+------+-----------------+
```

If the SELECT list includes an aggregate function and there is no GROUP BY clause, then all other columns in the SELECT list must be part of an aggregate function.

```
SELECT branchNo, COUNT(staffNo)
FROM Staff
GROUP BY branchNo;
```

Error:

```
SELECT branchNo, COUNT(staffNo)
FROM Staff;
```

```
SELECT branchNo, COUNT(staffNo)
FROM Staff
WHERE branchNo = 'B003';
```

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT branchNo, COUNT(staffNo)
                                        FROM Staff
                                        GROUP BY branchNo;
+----------+----------------+
| branchNo | COUNT(staffNo) |
+----------+----------------+
| B007     |              1 |
| B003     |              3 |
| B005     |              2 |
+----------+----------------+
```

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT branchNo, COUNT(staffNo)
                                        FROM Staff;
ERROR: 1140 (42000): In aggregated query without GROUP BY, expression #1 of SELECT list contains
nonaggregated column 'miu_db.Staff.branchNo'; this is incompatible with sql_mode=only_full_group_by
```

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT branchNo, COUNT(staffNo)
                                        FROM Staff
                                        WHERE branchNo = 'B003';
+----------+----------------+
| branchNo | COUNT(staffNo) |
+----------+----------------+
| B003     |              3 |
+----------+----------------+
```

There might be many different branchNo values in the table.

Which one should it display with the COUNT? That is ambiguous.

Hence, SQL throws an error.

Outline

- Introduction to SQL commands
- DDL
- DML
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

Subqueries

- Some SQL statements can have a SELECT embedded within them.
- The results of this inner SELECT statement (or subselect) are used in the outer statement to help determine the contents of the final result.
- A sub-select can be used in the WHERE and HAVING clauses of an outer SELECT statement, where it is called a subquery or nested query.
- Subselects may also appear in INSERT, UPDATE, and DELETE statements.

Subqueries

Set up two tables

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Subqueries

Set up two tables

```
MySQL  localhost:3306 ssl  miu_db  SQL
```

```
CREATE TABLE Branch (
branchNo VARCHAR(4) PRIMARY KEY,
street    VARCHAR(30),
city      VARCHAR(20),
postcode  VARCHAR(10)
);
```

```
Query OK, 0 rows affected (0.0299 sec)
```

```
MySQL  localhost:3306 ssl  miu_db  SQL
```

```
INSERT INTO Branch (branchNo, street, city, postcode)
VALUES
('B005', '22 Deer Rd', 'London', 'SW1 4EH'),
('B007', '16 Argyll St', 'Aberdeen', 'AB2 3SU'),
('B003', '163 Main St', 'Glasgow', 'G11 9QX'),
('B004', '32 Manse Rd', 'Bristol', 'BS99 1NZ'),
('B002', '56 Clover Dr', 'London', 'NW10 6EU');
```

```
Query OK, 5 rows affected (0.0104 sec)

Records: 5  Duplicates: 0  Warnings: 0
```

```
MySQL  localhost:3306 ssl  miu_db  SQL
```

```
ALTER TABLE Staff
ADD CONSTRAINT fk_staff_branch
FOREIGN KEY (branchNo)
REFERENCES Branch(branchNo);
```

```
Query OK, 6 rows affected (0.0245 sec)
```

Subqueries

1. Using a subquery with equality

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

List the staff who work in the branch at ‘163 Main St’.

```
MySQL  localhost:3306 ssl  miu_db  SQL
SELECT *
FROM Staff
WHERE branchNo = (SELECT branchNo
                  FROM Branch
                  WHERE street = '163 Main St');
+---------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+---------+-------+-------+------------+-----+------------+----------+----------+
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 24000.00 | B003     |
+---------+-------+-------+------------+-----+------------+----------+----------+
```

Subqueries

2. Using a Subquery to Retrieve Related Data

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

```
+---------+---------------+
| staffNo | branch_street |
+---------+---------------+
| SG14    | 163 Main St   |
| SG37    | 163 Main St   |
| SG5     | 163 Main St   |
| SL21    | 22 Deer Rd    |
| SL41    | 22 Deer Rd    |
| SA9     | 16 Argyll St  |
+---------+---------------+
```

Write an SQL query to display each staffNo, along with the street address of the branch where the staff member works.

```
MySQL  localhost:3306 ssl  miu_db  SQL  SELECT staffNo, (SELECT street
                                          FROM Branch
                                          WHERE Branch.branchNo = Staff.branchNo) AS branch_street
                                     FROM Staff;
+---------+---------------+
| staffNo | branch_street |
+---------+---------------+
| SG14    | 163 Main St   |
| SG37    | 163 Main St   |
| SG5     | 163 Main St   |
| SL21    | 22 Deer Rd    |
| SL41    | 22 Deer Rd    |
| SA9     | 16 Argyll St  |
+---------+---------------+
```

Subqueries

3. Using a subquery with aggregate function

```
+--------+-------+-------+------------+-----+------------+----------+----------+
| staffNo | fName | lName | position   | sex | DOB        | salary   | branchNo |
+--------+-------+-------+------------+-----+------------+----------+----------+
| SA9     | Mary  | Howe  | Assistant  | F   | 1970-02-19 | 90000.00 | B007     |
| SG14    | David | Ford  | Supervisor | M   | 1958-03-24 | 18000.00 | B003     |
| SG37    | Ann   | Beech | Assistant  | F   | 1960-11-10 | 12000.00 | B003     |
| SG5     | Susan | Brand | Manager    | F   | 1940-06-03 | 24000.00 | B003     |
| SL21    | John  | White | Manager    | M   | 1945-10-01 | 30000.00 | B005     |
| SL41    | Julie | Lee   | Assistant  | F   | 1965-06-13 |  9000.00 | B005     |
+--------+-------+-------+------------+-----+------------+----------+----------+
```

```
+--------------+
| AVG(salary)  |
+--------------+
| 30500.000000 |
+--------------+
```

List all staff whose salary is greater than the average salary, and show by how much their salary is greater than the average.

```
MySQL localhost:3306 ssl [redacted] miu_db SQL SELECT *
FROM Staff
WHERE salary > (SELECT AVG(salary) FROM Staff);
+--------+-------+-------+----------+-----+------------+----------+----------+
| staffNo | fName | lName | position  | sex | DOB        | salary   | branchNo |
+--------+-------+-------+----------+-----+------------+----------+----------+
| SA9     | Mary  | Howe  | Assistant | F   | 1970-02-19 | 90000.00 | B007     |
+--------+-------+-------+----------+-----+------------+----------+----------+
```

```
MySQL localhost:3306 ssl [redacted] miu_db SQL SELECT staffNo, fName, lName, position,
       salary - (SELECT AVG(salary) FROM Staff) AS salDiff
FROM Staff
WHERE salary > (SELECT AVG(salary) FROM Staff);
+--------+-------+-------+----------+-------------+
| staffNo | fName | lName | position  | salDiff     |
+--------+-------+-------+----------+-------------+
| SA9     | Mary  | Howe  | Assistant | 59500.000000 |
+--------+-------+-------+----------+-------------+
```

Subqueries

3. Using a subquery with aggregate function

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

```
+----------+--------------------+
| branchNo | highest_paid_staff |
+----------+--------------------+
| B002     | NULL               |
| B003     | Susan Brand        |
| B004     | NULL               |
| B005     | John White         |
| B007     | Mary Howe          |
+----------+--------------------+
```

Show each branch's ID along with the name of the highest-paid staff member in that branch.

```
MySQL  localhost:3306 ssl  miu_db  SQL
SELECT branchNo,
       (SELECT CONCAT(fName, ' ', lName)
        FROM Staff
        WHERE Staff.branchNo = Branch.branchNo
        ORDER BY salary DESC
        LIMIT 1) AS highest_paid_staff
FROM Branch;
+----------+--------------------+
| branchNo | highest_paid_staff |
+----------+--------------------+
| B002     | NULL               |
| B003     | Susan Brand        |
| B004     | NULL               |
| B005     | John White         |
| B007     | Mary Howe          |
+----------+--------------------+
```

Subqueries

3. Using a subquery with aggregate function

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

```
+----------+--------------------+
| branchNo | highest_paid_staff |
+----------+--------------------+
| B002     | NULL               |
| B003     | Susan Brand        |
| B004     | NULL               |
| B005     | John White         |
| B007     | Mary Howe          |
+----------+--------------------+
```

Show each branch's ID along with the name of the highest-paid staff member in that branch.

```
MySQL  localhost:3306 ssl  miu_db  SQL
SELECT b.branchNo,
       (SELECT *
        FROM Staff s
        WHERE s.branchNo = b.branchNo
        ORDER BY salary DESC
        LIMIT 1) AS highest_paid_staff
FROM Branch b;
```

Outline

- Introduction to SQL commands
- DDL
- DML
  - SELECT
  - Aggregates
  - Grouping
  - Subqueries
  - Join

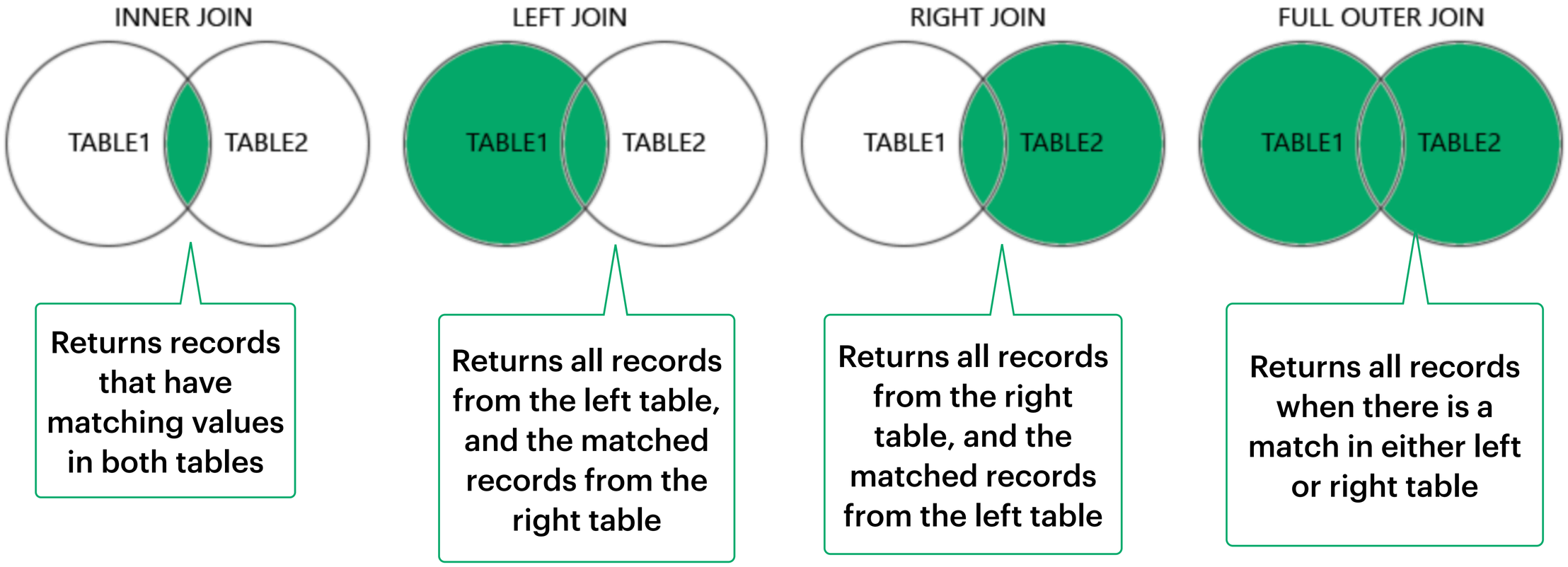
Multi-table Queries
JOINS in SQL

- Why Multi-table Queries?
  - A single table is not always enough to answer real-world queries.
  - Need to combine columns from multiple tables.
- What is a JOIN?
  - A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Multi-table Queries
JOINS in SQL

- How to Perform a JOIN?
  - Use multiple table names in the FROM clause.
  - Separate table names with a comma.
  - Use a WHERE clause to define how rows match.

Different Types of SQL JOINs



JOIN and INNER JOIN are functionally equivalent in SQL

Ref: https://www.w3schools.com/sql/sql_join.asp

Multi-table Queries

Create two tables: Client and Viewing

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	2013-05-24	too small
CR76	PG4	2013-04-20	too remote
CR56	PG4	2013-05-26	NULL
CR62	PA14	2013-05-14	no dining room
CR56	PG36	2013-04-28	NULL

Multi-table Queries

Create two tables: Client and Viewing

```
MySQL localhost:3306 ssl propertyagencydb SQL > CREATE TABLE Client (
                                             ->     clientNo VARCHAR(10) PRIMARY KEY,
                                             ->     fName VARCHAR(50),
                                             ->     lName VARCHAR(50),
                                             ->     telNo VARCHAR(20),
                                             ->     prefType VARCHAR(20),
                                             ->     maxRent INT,
                                             ->     eMail VARCHAR(100)
                                             -> );
Query OK, 0 rows affected (0.0107 sec)
MySQL localhost:3306 ssl propertyagencydb SQL > INSERT INTO Client (clientNo, fName, lName, telNo, prefType, maxRent, eMail) VALUES
                                             -> ('CR76', 'John', 'Kay', '0207-774-5632', 'Flat', 425, 'john.kay@gmail.com'),
                                             -> ('CR56', 'Aline', 'Stewart', '0141-848-1825', 'Flat', 350, 'astewart@hotmail.com'),
                                             -> ('CR74', 'Mike', 'Ritchie', '01475-392178', 'House', 750, 'mritchie01@yahoo.co.uk'),
                                             -> ('CR62', 'Mary', 'Tregear', '01224-196720', 'Flat', 600, 'maryt@hotmail.co.uk');
Query OK, 4 rows affected (0.0130 sec)

Records: 4  Duplicates: 0  Warnings: 0
MySQL localhost:3306 ssl propertyagencydb SQL > CREATE TABLE Viewing (
                                             ->     clientNo VARCHAR(10),
                                             ->     propertyNo VARCHAR(10),
                                             ->     viewDate DATE,
                                             ->     comment VARCHAR(100),
                                             ->     FOREIGN KEY (clientNo) REFERENCES Client(clientNo)
                                             -> );
Query OK, 0 rows affected (0.0098 sec)
MySQL localhost:3306 ssl propertyagencydb SQL > INSERT INTO Viewing (clientNo, propertyNo, viewDate, comment) VALUES
                                             -> ('CR56', 'PA14', '2013-05-24', 'too small'),
                                             -> ('CR76', 'PG4',  '2013-04-20', 'too remote'),
                                             -> ('CR56', 'PG4',  '2013-05-26', NULL),
                                             -> ('CR62', 'PA14', '2013-05-14', 'no dining room'),
                                             -> ('CR56', 'PG36', '2013-04-28', NULL);
Query OK, 5 rows affected (0.0028 sec)
```

Multi-table Queries

Simple JOIN (OR) INNER JOIN

Syntax

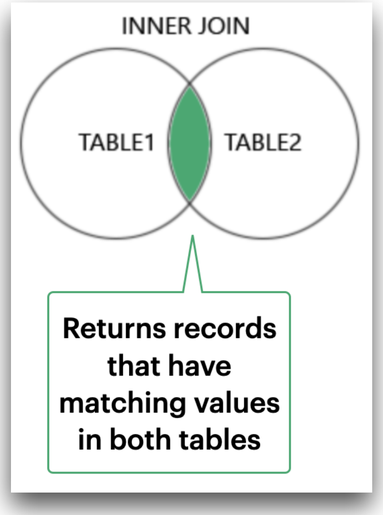
```
SELECT column_name(s)
FROM table1 t1, table2 t2
WHERE t1.column_name = t2.column_name;
```

Multi-table Queries

Simple JOIN (OR) INNER JOIN

Modern Alternative (ANSI JOIN Syntax)

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



Multi-table Queries

Simple JOIN

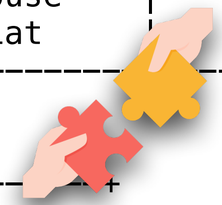
List the names of all clients who have viewed a property, along with any comments supplied.

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	2013-05-24	too small
CR76	PG4	2013-04-20	too remote
CR56	PG4	2013-05-26	NULL
CR62	PA14	2013-05-14	no dining room
CR56	PG36	2013-04-28	NULL



clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PA14	too small
CR76	John	Kay	PG4	too remote
CR56	Aline	Stewart	PG4	NULL
CR62	Mary	Tregear	PA14	no dining room
CR56	Aline	Stewart	PG36	NULL

Multi-table Queries

Simple JOIN: Modern Alternative (ANSI JOIN Syntax)

List the names of all clients who have viewed a property, along with any comments supplied.

```
MySQL  localhost:3306 ssl  propertyagencydb  SQL
```

```
SELECT c.clientNo, fName, lName, propertyNo, comment
FROM Client c
JOIN Viewing v ON c.clientNo = v.clientNo;
```

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PA14	too small
CR76	John	Kay	PG4	too remote
CR56	Aline	Stewart	PG4	NULL
CR62	Mary	Tregear	PA14	no dining room
CR56	Aline	Stewart	PG36	NULL

Multi-table Queries

Sorting a JOIN

Write an SQL query to retrieve the client number, first name, last name, property number, viewing date, and any comments for all clients who have viewed properties. The results should be sorted by client number and viewing date.

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	2013-05-24	too small
CR76	PG4	2013-04-20	too remote
CR56	PG4	2013-05-26	NULL
CR62	PA14	2013-05-14	no dining room
CR56	PG36	2013-04-28	NULL

clientNo	fName	lName	propertyNo	viewDate	comment
CR56	Aline	Stewart	PG36	2013-04-28	NULL
CR56	Aline	Stewart	PA14	2013-05-24	too small
CR56	Aline	Stewart	PG4	2013-05-26	NULL
CR62	Mary	Tregear	PA14	2013-05-14	no dining room
CR76	John	Kay	PG4	2013-04-20	too remote

Multi-table Queries

Sorting a JOIN

Write an SQL query to retrieve the client number, first name, last name, property number, viewing date, and any comments for all clients who have viewed properties. The results should be sorted by client number and viewing date.

```
MySQL  localhost:3306 ssl  propertyagencydb  SQL
```

```
SELECT c.clientNo, fName, lName, propertyNo, viewDate, comment
FROM Client c
JOIN Viewing v ON c.clientNo = v.clientNo
ORDER BY c.clientNo, viewDate;
```

```
+----------+-------+---------+------------+------------+----------------+
| clientNo | fName | lName   | propertyNo | viewDate   | comment        |
+----------+-------+---------+------------+------------+----------------+
| CR56     | Aline | Stewart | PG36       | 2013-04-28 | NULL           |
| CR56     | Aline | Stewart | PA14       | 2013-05-24 | too small      |
| CR56     | Aline | Stewart | PG4        | 2013-05-26 | NULL           |
| CR62     | Mary  | Tregear | PA14       | 2013-05-14 | no dining room |
| CR76     | John  | Kay     | PG4        | 2013-04-20 | too remote     |
+----------+-------+---------+------------+------------+----------------+
```

Multi-table Queries

Sorting a JOIN (or) INNER JOIN

Write an SQL query to retrieve the client number, first name, last name, property number, viewing date, and any comments for all clients who have viewed properties. The results should be sorted by client number and viewing date.

```
MySQL  localhost:3306 ssl  propertyagencydb  SQL
SELECT c.clientNo, fName, lName, propertyNo, viewDate, comment
FROM Client c
INNER JOIN Viewing v ON c.clientNo = v.clientNo
ORDER BY c.clientNo, viewDate;
```

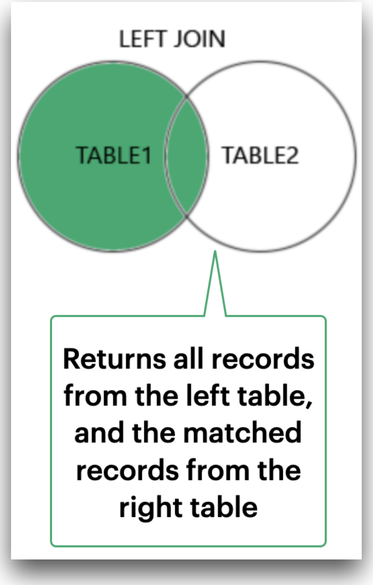
clientNo	fName	lName	propertyNo	viewDate	comment
CR56	Aline	Stewart	PG36	2013-04-28	NULL
CR56	Aline	Stewart	PA14	2013-05-24	too small
CR56	Aline	Stewart	PG4	2013-05-26	NULL
CR62	Mary	Tregear	PA14	2013-05-14	no dining room
CR76	John	Kay	PG4	2013-04-20	too remote

Multi-table Queries

Outer Joins - LEFT JOIN

Syntax

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



Multi-table Queries

Outer Joins - LEFT JOIN

You work for PropertyMatch, a real estate company. Your manager wants to analyze client engagement.

manager asks:

"Can you give me a list of all our clients, whether or not they have viewed any properties?

I want to see their names, what property (if any) they've visited, the viewing date, and any feedback they have given.

Make sure it's sorted nicely by client number and the date they visited."

Multi-table Queries

Outer Joins - LEFT JOIN

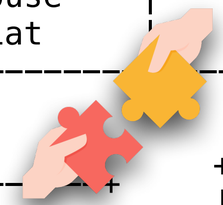
List of all our clients, whether or not they have viewed any properties.
Show their names, what property (if any) they have visited, the viewing date, and any feedback they have given.
Make sure it's sorted nicely by client number and the date they visited.

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	2013-05-24	too small
CR76	PG4	2013-04-20	too remote
CR56	PG4	2013-05-26	NULL
CR62	PA14	2013-05-14	no dining room
CR56	PG36	2013-04-28	NULL



clientNo	fName	lName	propertyNo	viewDate	comment
CR56	Aline	Stewart	PG36	2013-04-28	NULL
CR56	Aline	Stewart	PA14	2013-05-24	too small
CR56	Aline	Stewart	PG4	2013-05-26	NULL
CR62	Mary	Tregear	PA14	2013-05-14	no dining room
CR74	Mike	Ritchie	NULL	NULL	NULL
CR76	John	Kay	PG4	2013-04-20	too remote

Multi-table Queries

Outer Joins - LEFT JOIN

List of all our clients, whether or not they have viewed any properties.
Show their names, what property (if any) they have visited, the viewing date, and any feedback they have given.
Make sure it's sorted nicely by client number and the date they visited.

```
MySQL  localhost:3306 ssl  propertyagencydb  SQL
```

```
SELECT c.clientNo, fName, lName, propertyNo, viewDate, comment
FROM Client c
LEFT JOIN Viewing v ON c.clientNo = v.clientNo
ORDER BY c.clientNo, viewDate;
```

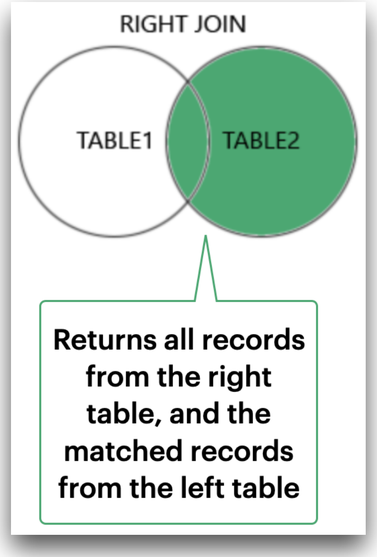
```
+----------+-------+---------+------------+------------+----------------+
| clientNo | fName | lName   | propertyNo | viewDate   | comment        |
+----------+-------+---------+------------+------------+----------------+
| CR56     | Aline | Stewart | PG36       | 2013-04-28 | NULL           |
| CR56     | Aline | Stewart | PA14       | 2013-05-24 | too small      |
| CR56     | Aline | Stewart | PG4        | 2013-05-26 | NULL           |
| CR62     | Mary  | Tregear | PA14       | 2013-05-14 | no dining room |
| CR74     | Mike  | Ritchie | NULL       | NULL       | NULL           |
| CR76     | John  | Kay     | PG4        | 2013-04-20 | too remote     |
+----------+-------+---------+------------+------------+----------------+
```

Multi-table Queries

Outer Joins - RIGHT JOIN

Syntax

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



Multi-table Queries

Outer Joins - RIGHT JOIN

You work for PropertyMatch, a real estate company. Your manager wants to analyze client engagement.

manager asks:

Give me a full list of all property viewings, even if we don't have the client's personal details in our system anymore.

I want to see who viewed what, when they viewed it, and what they said. If a client's info is missing, that's a red flag.

Multi-table Queries

Outer Joins - RIGHT JOIN

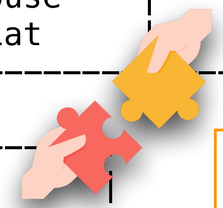
Give me a full list of all property viewings, even if we don't have the client's personal details in our system anymore. I want to see who viewed what, when they viewed it, and what they said. If a client's info is missing, that's a red flag.

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	2013-05-24	too small
CR56	PG36	2013-04-28	NULL
CR56	PG4	2013-05-26	NULL
CR62	PA14	2013-05-14	no dining room
CR76	PG4	2013-04-20	too remote
G1	PG10	2023-05-20	walk-in client



clientNo	propertyNo	viewDate	comment	clientNo
CR56	PA14	2013-05-24	too small	CR56
CR56	PG36	2013-04-28	NULL	CR56
CR56	PG4	2013-05-26	NULL	CR56
CR62	PA14	2013-05-14	no dining room	CR62
CR76	PG4	2013-04-20	too remote	CR76
G1	PG10	2023-05-20	walk-in client	NULL

Multi-table Queries

Outer Joins - RIGHT JOIN

Give me a full list of all property viewings, even if we don't have the client's personal details in our system anymore. I want to see who viewed what, when they viewed it, and what they said. If a client's info is missing, that's a red flag.

```
MySQL  localhost:3306 ssl  miu_db  SQL
```

```
SELECT v.*, c.clientNo
FROM Client c
RIGHT JOIN Viewing v
ON c.clientNo = v.clientNo;
```

clientNo	propertyNo	viewDate	comment	clientNo
CR56	PA14	2013-05-24	too small	CR56
CR56	PG36	2013-04-28	NULL	CR56
CR56	PG4	2013-05-26	NULL	CR56
CR62	PA14	2013-05-14	no dining room	CR62
CR76	PG4	2013-04-20	too remote	CR76
G1	PG10	2023-05-20	walk-in client	NULL

```
MySQL localhost:3306 ssl [redacted] miu_db SQL SELECT v.*, c.clientNo
                                               FROM Client c
                                               RIGHT JOIN Viewing v
                                               ON c.clientNo = v.clientNo
                                               WHERE c.clientNo IS NULL;
+----------+------------+------------+----------------+----------+
| clientNo | propertyNo | viewDate   | comment        | clientNo |
+----------+------------+------------+----------------+----------+
| G1       | PG10       | 2023-05-20 | walk-in client | NULL     |
+----------+------------+------------+----------------+----------+
1 row in set (0.0035 sec)
```

```
MySQL localhost:3306 ssl [redacted] miu_db SQL SELECT
                                                   v.*,
                                                   IF(c.clientNo IS NULL, 'GUEST', 'REGISTERED') AS client_type
                                               FROM Client c
                                               RIGHT JOIN Viewing v ON c.clientNo = v.clientNo;
+----------+------------+------------+----------------+-------------+
| clientNo | propertyNo | viewDate   | comment        | client_type |
+----------+------------+------------+----------------+-------------+
| CR56     | PA14       | 2013-05-24 | too small      | REGISTERED  |
| CR56     | PG36       | 2013-04-28 | NULL           | REGISTERED  |
| CR56     | PG4        | 2013-05-26 | NULL           | REGISTERED  |
| CR62     | PA14       | 2013-05-14 | no dining room | REGISTERED  |
| CR76     | PG4        | 2013-04-20 | too remote     | REGISTERED  |
| G1       | PG10       | 2023-05-20 | walk-in client | GUEST       |
+----------+------------+------------+----------------+-------------+
```

Multi-table Queries

Outer Joins - FULL JOIN (Or) FULL OUTER JOIN

Syntax

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

Multi-table Queries

Outer Joins - FULL JOIN (Or) FULL OUTER JOIN

Show me all client records, all viewings, and how they relate.

Even if a client hasn't viewed any properties, or if a viewing record has no matching client — I want to see it all.

```
1 SELECT * FROM public.viewing
2 ORDER BY viewid ASC
```

Data Output | Messages | Notifications

	viewid [PK] integer	clientno character varying (10)	propertyno character varying (10)	viewdate date	comment character varying (100)
1	1	CR56	PA14	2013-05-24	too small
2	2	CR76	PG4	2013-04-20	too remote
3	3	CR56	PG4	2013-05-26	[null]
4	4	CR62	PA14	2013-05-14	no dining room
5	5	CR56	PG36	2013-04-28	[null]

```
1 SELECT * FROM public.client
2 ORDER BY clientno ASC
```

Data Output | Messages | Notifications

	clientno [PK] character varying (10)	fname character varying (50)	lname character varying (50)	telno character varying (20)	preftype character varying (20)	maxrent integer	email character varying (100)
1	CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
2	CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk
3	CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
4	CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com

```sql
SELECT c.clientNo, c.fName, c.lName, v.viewDate, v.comment
FROM CLIENT c
FULL JOIN VIEWING v
ON c.clientNo = v.clientNo
```

Data Output | Messages | Notifications

	clientno character varying (10)	fname character varying (50)	lname character varying (50)	viewdate date	comment character varying (100)
1	CR56	Aline	Stewart	2013-05-24	too small
2	CR76	John	Kay	2013-04-20	too remote
3	CR56	Aline	Stewart	2013-05-26	[null]
4	CR62	Mary	Tregear	2013-05-14	no dining room
5	CR56	Aline	Stewart	2013-04-28	[null]
6	CR74	Mike	Ritchie	[null]	[null]