Java Spring & AWS
Sep 7, 2021 - Oct 8, 2021
Monday to Friday
9:30 AM ET - 4:30 PM ET

MySQL – Day 1 Training Plan

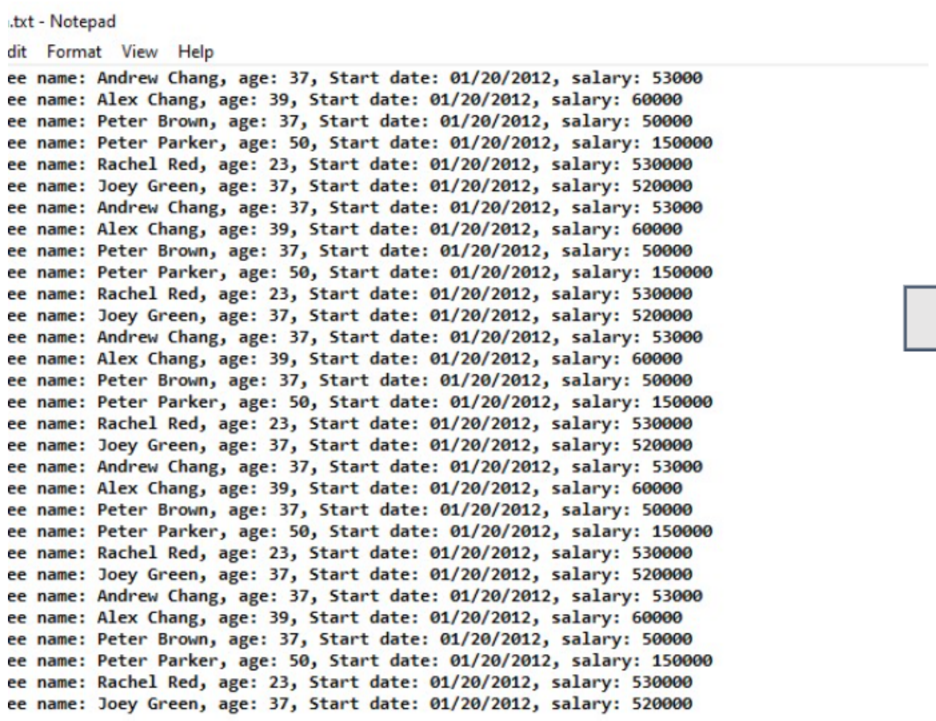
genda: Day 1: Introduction to MySQL

- Introduction and MySQL concepts
  - Why to use Database?
  - What is ER (Entity-relationship) – Modeling?
  - Normalization
  - Keys in SQL
  - Database Constraints
- MySQL Installation
- Data types
- Sub Languages in MySQL
  - Data Definition Language (DDL)
    - Create, Alter, Drop
  - Data Manipulation Language (DML)
    - Insert,Update,Delete
  - Data Query Language (DQL)
    - Select,Show,Help
  - Data Control Language (DCL)
    - Grant and Revoke
- Basic SQL
  - Conditional Staten
  - Comparison Opera
- Joins
  - Inner
  - Outer Join
  - Left Join
  - Right Join
  - Cross Join

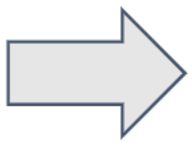
Data.txt - Notepad

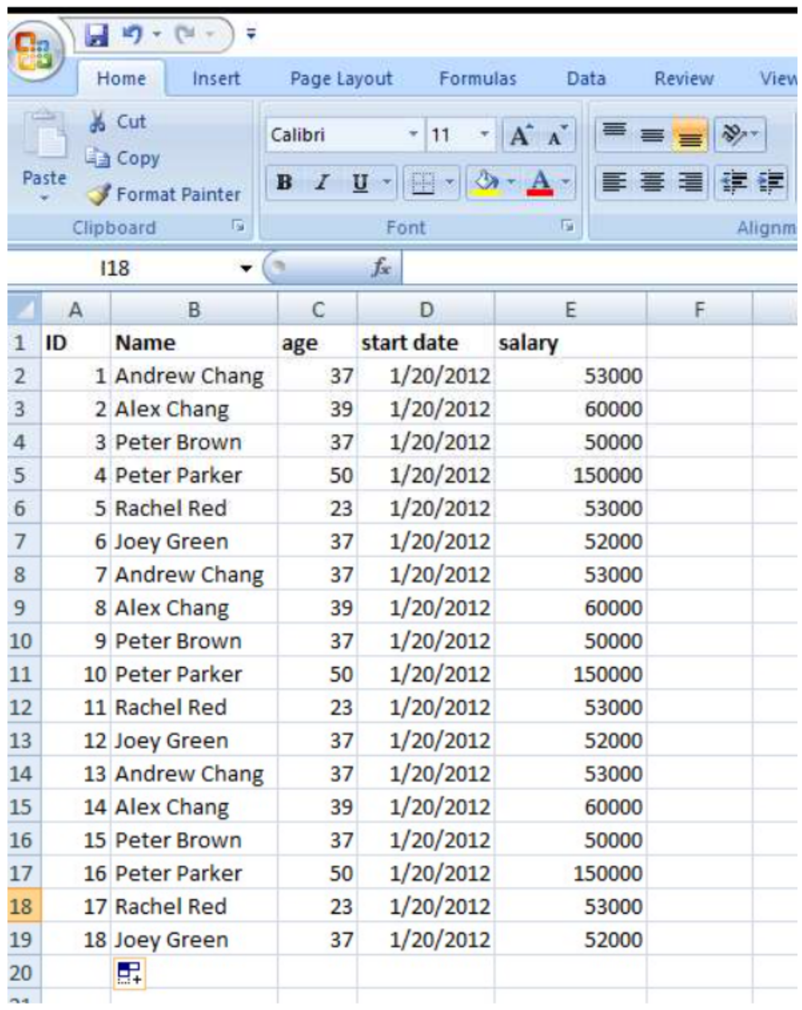
File Edit Format View Help

```
Employee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
Employee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
Employee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
Employee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
Employee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
Employee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
Employee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
Employee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
Employee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
Employee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
Employee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
Employee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
Employee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
Employee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
Employee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
Employee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
Employee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
Employee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
Employee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
Employee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
Employee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
Employee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
Employee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
Employee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
Employee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
Employee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
Employee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
Employee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
Employee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
Employee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
```

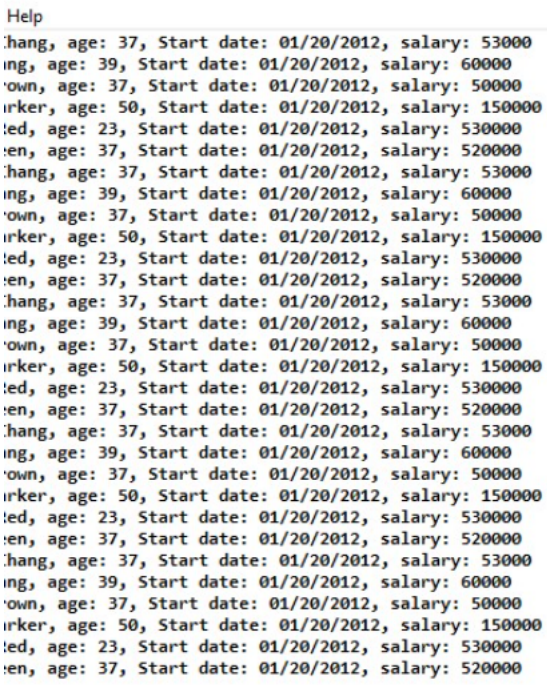


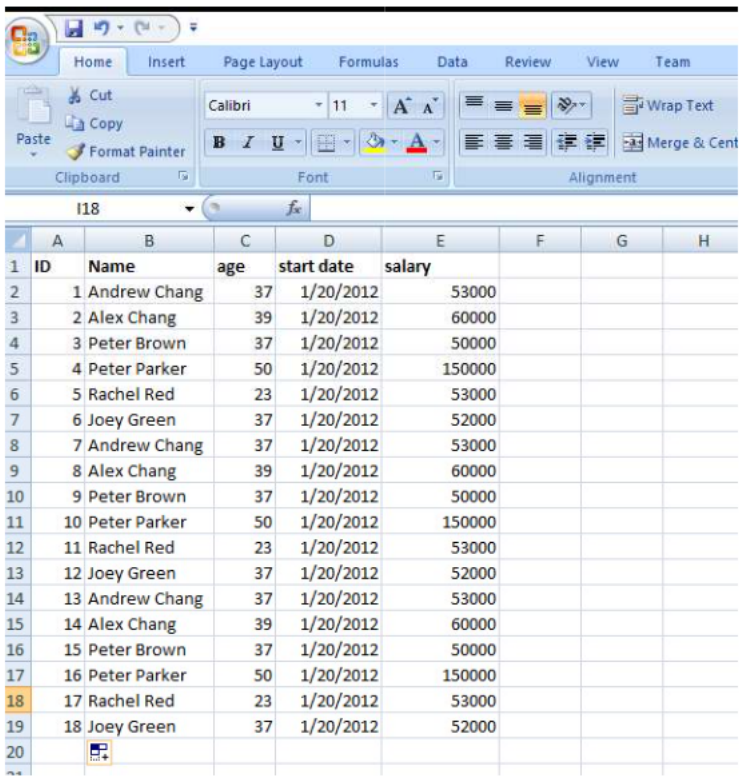
```
.txt - Notepad
dit  Format  View  Help
ee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
ee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
ee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
ee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
ee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
ee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
ee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
ee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
ee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
ee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
ee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
ee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
ee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
ee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
ee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
ee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
ee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
ee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
ee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
ee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
ee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
ee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
ee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
ee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
ee name: Andrew Chang, age: 37, Start date: 01/20/2012, salary: 53000
ee name: Alex Chang, age: 39, Start date: 01/20/2012, salary: 60000
ee name: Peter Brown, age: 37, Start date: 01/20/2012, salary: 50000
ee name: Peter Parker, age: 50, Start date: 01/20/2012, salary: 150000
ee name: Rachel Red, age: 23, Start date: 01/20/2012, salary: 530000
ee name: Joey Green, age: 37, Start date: 01/20/2012, salary: 520000
```





	A	B	C	D	E	F
1	ID	Name	age	start date	salary	
2	1	Andrew Chang	37	1/20/2012	53000	
3	2	Alex Chang	39	1/20/2012	60000	
4	3	Peter Brown	37	1/20/2012	50000	
5	4	Peter Parker	50	1/20/2012	150000	
6	5	Rachel Red	23	1/20/2012	53000	
7	6	Joey Green	37	1/20/2012	52000	
8	7	Andrew Chang	37	1/20/2012	53000	
9	8	Alex Chang	39	1/20/2012	60000	
10	9	Peter Brown	37	1/20/2012	50000	
11	10	Peter Parker	50	1/20/2012	150000	
12	11	Rachel Red	23	1/20/2012	53000	
13	12	Joey Green	37	1/20/2012	52000	
14	13	Andrew Chang	37	1/20/2012	53000	
15	14	Alex Chang	39	1/20/2012	60000	
16	15	Peter Brown	37	1/20/2012	50000	
17	16	Peter Parker	50	1/20/2012	150000	
18	17	Rachel Red	23	1/20/2012	53000	
19	18	Joey Green	37	1/20/2012	52000	
20						





	A	B	C	D	E
1	ID	Name	age	start date	salary
2	1	Andrew Chang	37	1/20/2012	53000
3	2	Alex Chang	39	1/20/2012	60000
4	3	Peter Brown	37	1/20/2012	50000
5	4	Peter Parker	50	1/20/2012	150000
6	5	Rachel Red	23	1/20/2012	53000
7	6	Joey Green	37	1/20/2012	52000
8	7	Andrew Chang	37	1/20/2012	53000
9	8	Alex Chang	39	1/20/2012	60000
10	9	Peter Brown	37	1/20/2012	50000
11	10	Peter Parker	50	1/20/2012	150000
12	11	Rachel Red	23	1/20/2012	53000
13	12	Joey Green	37	1/20/2012	52000
14	13	Andrew Chang	37	1/20/2012	53000
15	14	Alex Chang	39	1/20/2012	60000
16	15	Peter Brown	37	1/20/2012	50000
17	16	Peter Parker	50	1/20/2012	150000
18	17	Rachel Red	23	1/20/2012	53000
19	18	Joey Green	37	1/20/2012	52000
20					

Introduction to MySQL concepts

hy to use databases?

- A database is a collection of information that is organized so th easily be accessed, managed, and updated.
- Benefits of using database:
  - To eliminate or reduce the data Redundancy
  - To remove or reduce the data Inconsistency
  - For data Standardization
  - Secure all the data in one shell. Performed Role based secu
  - To make sure that the Integrity of the data is maintained

Introduction to MySQL concepts (cont

hat is ER (Entity-relationship) – Modeling?

- **ER Diagram** is a visual representation of data that describes hov related to each other.
- In ER Model, we disintegrate data into **entities**, **attributes** and s **relationships** between entities, all this can be represented visu ER diagram.
- ER Diagrams are most often used to design or debug relational the fields of software engineering.
- they use a defined set of symbols such as rectangles, diamonds connecting lines to depict the interconnectedness of entities, re and their attributes.

Introduction to MySQL concepts (cont

mponents of ER Diagram

- Entity, Attributes, Relationships etc form the components of ER Diagram and there a symbols and shapes to represent each one of them.
- Let's see how we can represent these in our ER Diagram.

Entity

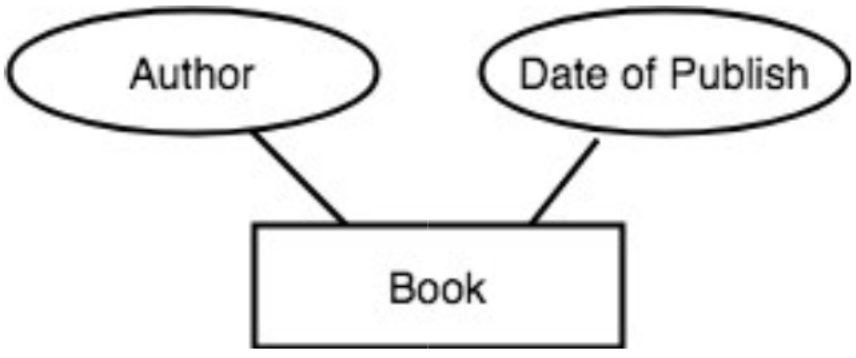
Simple rectangular box represents an Entity.

Introduction to MySQL concepts (cont

mponents of ER Diagram

Attributes for any Entity

Ellipse is used to represent attributes of any entity. It is connec
entity



Introduction to MySQL concepts (cont

Diagram: Relationship

Relationship describes relation between entities. Relationship is rep
ing diamonds or rhombus.

Introduction to MySQL concepts (cont

Diagram: Binary Relationship Binary Relationship means relation b
tities. This is further divided into three types.

ne to One Relationship: This type of relationship is rarely seen in rea

ne to Many Relationship: The below example showcases this relatic
eans that 1 student can opt for many courses, but a course can only
udent. Sounds weird! This is how it is.

Introduction to MySQL concepts (cont

R Diagram: Binary Relationship

any to One Relationship: It reflects business rule that many entities sociated with just one entity. For example, Student enrolls for only ıt a Course can have many Students.

any to One Relationship: The below diagram represents that one st ıroll for more than one courses. And a course can have more than 1 ırolled in it

Keys

Keys are important part of the arrangement of a table. Keys make uniquely identify a table's each part or record of a field or combin fields.

- Type of keys:
  - Primary key
  - Natural Key
  - Surrogate/Artificial key
  - Candidate keys
  - Secondary or Alternative key
  - Composite Key
  - Foreign Key

Keys (cont.)

Primary key: The candidate key which is very suitable to be the main key of tal key.

The primary keys are compulsory in every table.

The properties of a primary key are:

- Model stability
- Occurrence of minimum fields
- Defining value for every record i.e. being definitive
- Feature of accessibility

Student Id	First name of student	Last name of student	Course Id
123456	Jasmine	Shaik	001
123457	Rose	Mary	002
123458	Lily	Holmes	003

Primary key

Keys (cont.)

Natural Key: It is a key that is naturally declared as the Primary key. Natural ke sometimes called as business or domain keys because these key are based on t observation. So it is a key whose attributes or values exists in the real world. Th have logical relationship with the table.

- For Example: Social Security Number (SSN) is a natural key that can be dec primary key.

Surrogate/Artificial key : Surrogate key is artificially generated key and its mair be the primary key of table. Artificial keys do not have meaning to the table. Th properties of surrogate or artificial keys.

- They are unique because these just created when you don't have any natu
- They are integer values. One cannot find the meaning of surrogate keys in
- End users cannot surrogate key.Surrogate keys are allowed when no prope parameter of primary key.
- The primary key is huge and complex.
- Example: Table which has the details of the student has primary key but it complex. The addition of row id column to it is the DBA's decision, where t is row id.

Keys (cont.)

Candidate keys are the set of fields; primary key can be selected from these fie properties or attributes acts as a primary key for a table. Every table must have candidate key or several candidate keys.

- For example: The fields of a candidate key uniquely identify a student. It ha like – Being unique and Parameter of irreducibility.

Student Id	First name of student	Last name of student	Course Id
123456	Jasmine	Shaik	001
123457	Rose	Mary	002
123458	Lily	Holmes	003

Candidate keys

Keys (cont.)

Secondary or Alternative key: The rejected candidate keys as primary keys are secondary or alternative keys.

Student Id	First name of student	Last name of student	Course Id
123456	Jasmine	Shaik	001
123457	Rose	Mary	002
123458	Lily	Holmes	003

Secondary or Alternative keys

Keys (cont.)

Composite Key has two or more properties which specially identifies the occur entity.

StudentId	StudentName	Year	Semester
0023765	John Doe	2009	2
0035643	Ann Smith	2009	2
0061234	Pete	2009	2

Composite Key

StudentId	UnitCode	UnitName
0023765	UG45783	Advance Database
0023765	UG45832	Network Systems
0023765	UG45734	Multi-User Operating Systems
0035643	UG45832	Network Systems
0035643	UG45951	Project
0061234	UG45783	Advance Database

Tables in
First Normal Form

Keys (cont.)

- Foreign key creates a link between tables. It references the prima another table and links it.
- A foreign key can be a common key in two database table

Student

	id	Fullname	Course_id	Phone_number
▶	1	Rachel Green	1	62998047
	2	Monica Geller	1	1234567
	3	Phoebe Buffay	2	6399136220
	4	Joey Tribbiani	3	4595165911
	5	Chandler Bing	2	2778421170
	6	Ross Geller	4	8106608394
	7	Brad Pitt	3	4197778741

Foreign key

Primary key

Course

id	Title
1	Python
2	Machine Learning
3	Django
4	AWS

What is Normalization? 1NF, 2NF, 3NF and BCN

base normalization (or normalization) is the process of orga
butes) and tables (relations) of a relational database to minimize dat

UNF	1NF	2NF	
All the attributes of the database are simply listed, without any sense of order or grouping.	• It has an identifying key • Each table cell should contain a single value.	• It is in 1NF • All non-key attributes are functionally dependent on the whole key (not part of the key	• I • I t c

Normalization(Example)

normalised Form

oject de	Project Title	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	De Na
010	Pensions System	M Phillips	24500	S10001	A Smith	L004	
010	Pensions System	M Phillips	24500	S10030	L Jones	L023	
010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	
045	Salaries System	H Martin	17400	S10010	B Jones	L004	
045	Salaries System	H Martin	17400	S10001	A Smith	L004	
045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	
045	Salaries System	H Martin	17400	S13210	W Richards	L008	
064	HR System	K Lewis	12250	S31002	T Gilbert	L028	
064	HR System	K Lewis	12250	S21010	P Lewis	L004	
064	HR System	K Lewis	12250	S10034	B James	L009	
010	Pensions System	M Phillips	24500	S10001	A Smith	L004	
				S10030	L Jones	L023	
				S21010	P Lewis	L004	
045	Salaries System	H Martin	17400	S10010	B Jones	L004	
				S10001	A Smith	L004	
				S31002	T Gilbert	L028	
				S13210	W Richards	L008	
064	HR System	K Lewis	12250	S31002	T Gilbert	L028	
				S21010	P Lewis	L004	
				S10034	B James	L009	

Normalization(Example)

First Normal Form(1NF)

- The rule is: remove any repeating attributes to a new table. The process is as follows:

Project Code	Project Title	Project Manager	Project Budget
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

Project Code	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	S10001	A Smith	L004	IT	22.0
PC010	S10030	L Jones	L023	Pensions	18.5
PC010	S21010	P Lewis	L004	IT	21.0
PC045	S10010	B Jones	L004	IT	21.7
PC045	S10001	A Smith	L004	IT	18.0
PC045	S31002	T Gilbert	L028	Database	25.5
PC045	S13210	W Richards	L008	Salary	17.0
PC064	S31002	T Gilbert	L028	Database	23.2
PC064	S21010	P Lewis	L004	IT	17.5
PC064	S10034	B James	L009	HR	16.5

Key

Depends on part of the key

Normalization(Example)

Second Normal Form(2NF)

- The rule is: remove any non-key attributes that only depend on part of the tabl
- A transitive functional dependency is when changing a non-key column, might non-key columns to change

	Project Title	Project Manager	Project Budget
	Pensions System	M Phillips	24500
	Salaries System	H Martin	17400
	HR System	K Lewis	12250

	Employee No.	Hourly Rate
	S10001	22.00
	S10030	18.50
	S21010	21.00
	S10010	21.75
	S10001	18.00
	S31002	25.50
	S13210	17.00
	S31002	23.25
	S21010	17.50
	S10034	16.50

Employee No.	Employee Name	Department No.	Department Name
S10001	A Smith	L004	IT
S10030	L Jones	L023	Pensions
S21010	P Lewis	L004	IT
S10010	B Jones	L004	IT
S31002	T Gilbert	L028	Database
S13210	W Richards	L008	Salary
S10034	B James	L009	HR

(1) Change in Employee Name may change in Deparment Name

It is a transitive functional dependency

(2) Change
Deparmen

lormalization(Example)

Third Normal Form(3NF)

- The rule is: remove to a new table any non-key attributes that are more de key attributes than the table key

Project

Project Code	Project Title	Project Manager	Project Budget
PC010	Pensions System	M Phillips	24500
PC045	Salaries System	H Martin	17400
PC064	HR System	K Lewis	12250

Project Team

Project Code	Employee No.	Hourly Rate
PC010	S10001	22.00
PC010	S10030	18.50
PC010	S21010	21.00
PC045	S10010	21.75
PC045	S10001	18.00
PC045	S31002	25.50
PC045	S13210	17.00
PC064	S31002	23.25
PC064	S21010	17.50
PC064	S10034	16.50

Employee

Employee No.	Employee Name	Departm No. *
S10001	A Smith	L004
S10030	L Jones	L023
S21010	P Lewis	L004
S10010	B Jones	L004
S31002	T Gilbert	L023
S13210	W Richards	L008
S10034	B James	L000

Department No.	Department Name
L004	IT
L023	Pensions
L028	Database
L008	Salary
L009	HR

Departn

3NF: Non-Key Dependencies Removed

BCNF (3.5)

BCNF (Boyce-Codd Normal Form) does **not allow dependencies attributes that belong to candidate keys**.

BCNF is a refinement of the third normal form in which it drops the key attribute from the 3rd normal form.

r Example: following is 3NF but not NF, since

Subject column is a prime attribute and *professor* is a non-prime attribute,

Subject is dependent on *professor* (since one professor teaches only one subject, but one subject may have two different professors.)

Primary key

student_id	subject
101	Java
101	C++
102	Java
103	C#
104	Java

BCNF

For Example: From 3NF to BCNF.

	subject	professor
	Java	P.Java
	C++	P.Cpp
	Java	P.Java2
	C#	P.Chash
	Java	P.Java

Student Table

student_id
101
101
and so on...

Professor Table

p_id	professor
1	P.Java
2	P.Cpp
and so on...	

Introduction to MySQL concepts (cont

ySQL Installation

- **For Windows**

  Refer to installation guide:
  MySQL all required features installation on Windows.pdf

- **For Macs**

  Refer to installation guide:
  MySQL Server installation on MacOS.pdf
  MySQL Workbench (UI) installation on MacOS.pdf

  Or

  Follow the instruction here:
  https://dev.mysql.com/doc/refman/8.0/en/osx-installation-pkg.html

Database Constraints

SQL constraints are used to specify rules for the data in a table.
If there is any violation between the constraint and the data action, the action the constraint.
Constraints can be specified when the table is created (inside the CREATE TABL after the table is created (inside the ALTER TABLE statement).

Constraints can be defined in two ways

- 1. The constraints can be specified immediately after the column definitior column- level definition.
- 2. The constraints can be specified after all the columns are defined. This is level definition.

Types of constraints

- A NOT NULL constraint
- A unique key constraint (A unique constraint)
- A primary key constraint
- A foreign key constraint (referential constraint or a referential integrity cor

Introduction to MySQL concepts (cont

ample of constraints:

```
create table Student (
    id int PRIMARY KEY,
    Fullname varchar(5) NOT NULL,
    course_id int not null,
    phone_number CHAR(2) UNIQUE,
    FOREIGN KEY(course_id) REFERENCES course(
);
```

Data types

Properly defining the fields in a table is important to the overall op
your database. You use only the type and size of field you really ne
MySQL uses many different data types broken into three categorie

- **Numeric:** INT , FLOAT
- **Date and Time:** DATE , DATETIME
- **String Types:** CHAR, VARCHAR() , VARCHAR2, BLOB or TEXT

Data types (cont.)

Numeric

- **INT** – A normal-sized integer that can be signed or unsigned. If signed, the is from -2147483648 to 2147483647. If unsigned, the allowable range is fr 4294967295. You can specify a width of up to 11 digits.
- **FLOAT(M,D)** – A floating-point number that cannot be unsigned. You can d display length (M) and the number of decimals (D). This is not required an 10,2, where 2 is the number of decimals and 10 is the total number of digi decimals). Decimal precision can go to 24 places for a FLOAT.
- Ex: if you insert 999.00009 into a FLOAT(7,4) column, the approximate res
- **DOUBLE(M,D)** – A double precision floating-point number that cannot be u can define the display length (M) and the number of decimals (D). This is n will default to 16,4, where 4 is the number of decimals. Decimal precision places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** – An unpacked floating-point number that cannot be unsig unpacked decimals, each decimal corresponds to one byte. Defining the di and the number of decimals (D) is required. NUMERIC is a synonym for DE

Data types (cont.)

Date and Time

- **DATE** – A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-3 December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** – A date and time combination in YYYY-MM-DD HH:MM:SS forr 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the af December 30th, 1973 would be stored as 1973-12-30 15:30:00.

Data types (cont.)

String types

- **CHAR(M)** – A fixed-length string between 1 and 255 characters in length (f CHAR(5)), right-padded with spaces to the specified length when stored. D is not required, but the default is 1.
- **VARCHAR(M)** – A variable-length string between 1 and 255 characters in le example, VARCHAR(25). You must define a length when creating a VARCHA

Sub Languages in MySQL

DDL (Data Definition Language): statements are used to define the database structure or schema

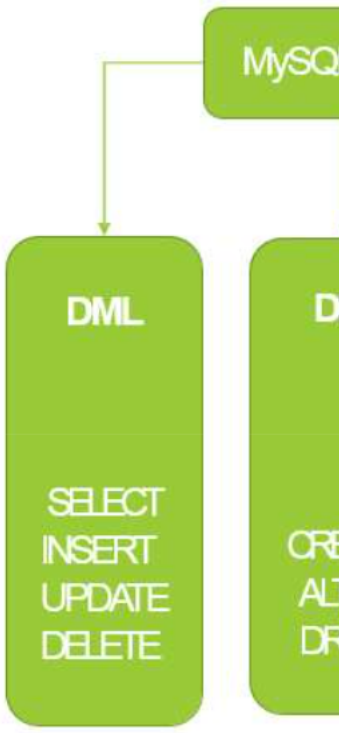
DML (Data Manipulation Language): statements are used for managing data within schema objects DML deals with data manipulation,

DCL (Data Control Language): DCL statements control the level of access that users have on database objects.

- **GRANT** – allows users to read/write on certain database objects
- **REVOKE** – keeps users from read/write permission on database objects

TCL (Transaction Control Language): statements allow you to control and manage transactions to maintain the integrity of data.

- **BEGIN Transaction** – opens a transaction
- **COMMIT Transaction** – commits a transaction
- **ROLLBACK Transaction** – ROLLBACK a transaction in case of any error



DDL Queries

ıta Definition Language (DDL) is a vocabulary used to define data structure

ıta Definition Language understanding with database schemas and describ
ı should consist in the database, therefore language statements like CREA
ER TABLE

DDL Queries - Create

CREATE DATABASE database_name

```
create database mySchool;
```

CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);

```
create table student(
    id int primary key,
    fullname varchar(5) not null,
    course_id int not null,
    phone_number char(9) unique
);
```

DDL Queries - Alter

_TER statements are used to modify the definition of existin

ER {DATABASE | SCHEMA} [db_name]
EFAULT] CHARACTER SET [=] charset_name
EFAULT] COLLATE [=] collation_name

```
efault MySQL character set and collation
latin1, latin1_swedish_ci)
TER DATABASE mySchool
ARACTER SET utf8
LATE utf8_general_ci;
```

ALTER TABLE table_name
MODIFY COLUMN column_name da

```
ALTER TABLE student
MODIFY COLUMN phone_numbe
```

DDL Queries – Drop

se DROP statements to remove existing entities.

OP DATABASE database_name

```
OP DATABASE mySchool;
```

DROP TABLE database_nam

```
DROP TABLE student;
```

DDL Queries - Truncate

ate table
ves all rows from a table or specified
ons of a table, without logging the
ual row deletions. The TRUNCATE
E command is used to delete complete
om an existing table.

an also use DROP TABLE command to
complete table but it would remove
ete table structure form the database
ou would need to re-create this table
again if you wish you store some data.

```
TRUNCATE [TABLE] tbl_
```

```
TRUNCATE TABLE
```

ta Manipulation Language (DML) Statem

- statements are used to work with the data IN tables.
- statements affect records in a table:
  - > Selecting a few records from a table
  - > Inserting new records,
  - > Deleting unnecessary records,
  - > Updating/modifying existing records.

SELECT – Access data in database (limited form of DML not changing da
INSERT – insert new records
UPDATE – update/Modify existing records
DELETE – delete existing records

ata Manipulation Language (DML) Stater

ne or more rows to a table or a view in database

INSERT INTO table_name VALUES (value1, value2, value3…)

INSERT INTO table_name (column2, column5,column6) VALUES (value

For example:

INSERT INTO TraineeList VALUES (1, ‘John’ , ‘Smith’)

INSERT INTO TraineeList (FirstName,LastName) VALUES (‘Sara’ , ‘Wilson’)

Data Manipulation Language (DML) Statement

te:

ges existing data in a table or view

```
UPDATE table_name SET
column_name1 = new_value1,
column_name2 = new value2
WHERE some_column = some_value
(condition)
```

ple:

```
UPDATE TraineeList  SET FirstName = 'Philip'  WHERE Trainee_ID = 1
```

ata Manipulation Language (DML) State

:LETE : Removes one or more rows from a table or view

- DELETE * FROM table_name
- DELETE FROM table_name WHERE some_column = some_value (

mple:

- DELETE FROM TraineeList WHERE FirstName = ‘sara’

erences between Truncate and Delete Statem

TRUNCATE	DELETE
is a **DDL command**	DELETE is a **DML command**
is executed using a table lock and **whole table is** emove all records.	DELETE is executed using a row lock, **eac** deletion.
ollback after performing Truncate.	We **can Rollback** after performing DELETE
removes all rows from a table. (We cannot use se with TRUNCATE.)	The DELETE command is used to **remove WHERE condition. (** (We can use where c delete specific records.)
ging in transaction log, so it is performance wise faster.	It maintain the log, so it slower than TRUNC
TABLE removes the data by deallocating the data to store the table data and records only the page s in the transaction log.	The DELETE statement removes rows one in the transaction log for each deleted row
mn is reset to its seed value if table contains any mn.	Identity of column keep DELETE retain the
cate on a table you need at least ALTER permission .	To use Delete you need DELETE permissio
es the less transaction space than Delete statement.	Delete uses the more transaction space tha
nnot be used with indexed views	Delete can be used with indexed views
TABLE can't activate a trigger because the oes not log individual row deletions. When we run mmand to remove all rows of table then it actually oves any row, rather it deallocates the data pages. In ncate triggers will not be fired because no modification	**Delete activates a trigger** because the op When we execute Delete command, DELE present. Delete is a DML command and it basis from a table. Which means delete is r from the table. Triggers are fired when a DM

asic query Statements

CT :

ves rows from the database and enables the selection of one or many rows
ne or many tables in MySQL. The full syntax of the SELECT statement is c

```
CT select_list (* | column_name)
l table_name
RE search_condition  GROUP BY group_by_expression  HAVING search_
R BY order_expression ASC | DESC
```

```
SELECT * FROM student;
```

Query statements: Show, Help

: used to get more details about databases and tables.

Description	Command
List all databases on the sql server.	show databases;
To see all the tables in the db.	show tables;
Returns the columns and column information pertaining to the designated table.	show columns from [table nam

: The HELP statement returns online information from the MySQL Referer

ription	Comma
ntents to retrieve a list of the top-level help categories	HELP '
list of topics in a given help category, such as Data Types, use the category	HELP '
on a specific help topic, such as the ASCII() function or the CREATE TABLE statement, use ociated keyword or keywords:	HELP ' HELP '

Data Control Language (DCL)

ANT: Used to provide any user access privileges or other privileges for the

IT : Data Control Language(DCL) is used to control privileges in Data
ion in the database, such as for creating tables, sequences or views, a use

eate a user, then you can try to create new connection with this user and p

```
REATE USER 'thien'@'localhost' IDENTIFIED BY 'password';
```

amples of GRANT permission:

```
RANT ALL ON db1.* TO 'thien'@'localhost';
```

```
ANT SELECT, INSERT, DELETE, UPDATE ON student TO 'thien'@'l
```

Data Control Language (DCL)

EVOKE: Used to take back permissions from any user.

OKE ALL PRIVILEGES, GRANT OPTION FROM user_or_role [, user_or
OKE 'role1', 'role2' FROM 'user1'@'localhost', 'user2'@'localhost';

xample:

```
KE ALL PRIVILEGES, GRANT OPTION FROM 'thien'@'localhost';
```

2	18:18:14	select * from student LIMIT 0, 1000

Code: 1142. SELECT command denied to user 'thien'@'localhost' for table student

Basic SQL – Where Clause

nditional Statements: WHERE Clause

t statements:

RE Clause: Specify an actual value as condition to filter the SELECT state

CT columnName FROM tableName WHERE condition;

kample:

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C1	United State	1001
	C2	Canada	1002
	C3	Astralia	1002

```
T COUNTRY_NAME FROM Countries WHERE REGION_ID=1001;
```

	COUNTRY_NAME
▶	United State

Basic SQL - Sorting

g: When you select rows, the MySQL server is free to return them in
t it otherwise by saying how to sort the result.

CT field1, field2,...fieldN table_name1, table_name2...
R BY field1, [field2...]
[DESC]]

- You can sort the returned result on any field, if that field is being listec
- You can sort the result on more than one field.
- You can use the keyword ASC or DESC to get result in ascending or default, it's the ascending order.
- You can use the WHERE...LIKE clause in the usual way to put a conc

Basic SQL: ORDER BY Example

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C1	United State	1001
	C2	Canada	1002
	C3	Astralia	1002

```
SELECT * FROM Countries ORDER BY COUNTRY_NAME;
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C3	Astralia	1002
C2	Canada	1002
C1	United State	1001

```
SELECT * FROM Countries ORDER BY COUNTRY_NAME DESC;
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C1	United State	1001
C2	Canada	1002
C3	Astralia	1002

Basic SQL: Group BY Example

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C1	United State	1001
	C4	Vietnam	NULL
	C5	Indonesia	NULL
	C2	England	1002
	C6	Brazil	1003
	C3	Netherland	1002
	C7	Poland	1004
	C8	Cananda	1001

```
select region_id,count(*) from countries group by region_id;
```

	region_id	count(*)
▶	1001	2
	NULL	2
	1002	2
	1003	1
	1004	1

Logical Operators

)L supports the following logical operations :

- ➢ AND(&&) Operator
- ➢ OR(||) Operator
- ➢ NOT(!) Operator

)L AND(&&) Operator : The logical AND(&&) operator indicates whether th
ue. Lets see a statement using AND operator.

```
mysql> select studid, name from student where marks > 80
and marks < 100;
                                (or)
mysql> select studid, name from student where marks > 80
&& marks < 100;
+---------+--------+
| studid | name   |
+---------+--------+
|       4 | jack   |
|       8 | mille |
+---------+--------+
2 rows in set (0.00 sec)
```

above example it will list the studid and name of the student who have sec
d less than 100.

Logical Operators

)L OR(||) Operator :

)gical OR(||) operator indicates whether either operand is true. Lets see a s)erator.

```
mysql> select name, marks, address from student where
name like 'a%' or name like 's%';
                                        (or)
mysql> select name, marks, address from student where
name like 'a%' || name like 's%';
+-------+-------+--------------------+
| name  | marks | address            |
+-------+-------+--------------------+
| steve |   100 | 5th cross street   |
| anne  |   100 | downing street     |
| steve |    75 | downing street     |
| anne  |    80 | edinburgh          |
+-------+-------+--------------------+
4 rows in set (0.00 sec)
```

above statement it will list the name, marks and address of the student wh)e letter A and S.

Logical Operators

QL NOT(!) Operator : The logical NOT(!) operator have only one operand a
ge of the value.

```
mysql> select * from student where not (studid=1);
                                  (or)
mysql> select * from student where ! (studid=1);
+---------+-------+-------+------------------+----------+
| studid | name   | marks | address          | phone    |
+---------+-------+-------+------------------+----------+
|      2 | david |   100 | welling street   |  547896 |
|      4 | jack  |    82 | welling street   | 2436821 |
|      5 | anne  |   100 | downing street   | 2634821 |
|      6 | steve |    75 | downing street   | 2874698 |
|      7 | anne  |    80 | edinburgh        | 2569843 |
|      8 | mille |    98 | victoria street | 1236547 |
+---------+-------+-------+------------------+----------+
6 rows in set (0.00 sec)
```

list all the student details except the studid 1.

Comparison Operators

on operators are used in the WHERE clause to determine which records to select. Here that you can use in MySQL:

Comparison Operator	Description
=	Equal
<=>	Equal (Safe to compare NULL va
<>	Not Equal
!=	Not Equal
>	Greater Than
>=	Greater Than or Equal
<	Less Than
<=	Less Than or Equal
IN ( )	Matches a value in a list
NOT	Negates a condition
BETWEEN	Within a range (inclusive)
IS NULL	NULL value
IS NOT NULL	Non-NULL value
LIKE	Pattern matching with % and _

Comparison Operators: Equality Operator

lySQL, you can use the = operator to test for equality in a query. The = ope
ality with values that are not NULL.

example:

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C1	United State	1001
C4	Vietnam	NULL
C5	Indonesia	NULL
C2	England	1002
C3	Germany	1002

```
LECT * FROM Countries WHERE region_id = 1002;
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C2	England	1002
C3	Germany	1002

Comparison Operators: Inequality Operator

QL, you can use the <> or != operators to test for inequality in a query.
ample, we could test for inequality using the <> operator, as follows:

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C1	United State	1001
C4	Vietnam	NULL
C5	Indonesia	NULL
C2	England	1002
C3	Germany	1002

```
 * FROM Countries WHERE COUNTRY_NAME != 'England';
 * FROM Countries WHERE COUNTRY_NAME <> 'England';
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C1	United State	1001
C4	Vietnam	NULL
C5	Indonesia	NULL
C3	Germany	1002

f these queries would return the same results.

Comparison Operators: Comparison Operator

can use the < operator in MySQL to test for an expression less than.

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C1	United State	1001
C4	Vietnam	NULL
C5	Indonesia	NULL
C2	England	1002
C3	Germany	1002

```
.ECT * FROM Countries WHERE region_id < 1002;
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C1	United State	1001

Comparison Operators: IN Condition

SQL IN condition is used to help reduce the need to use multiple OR condi
T, INSERT, UPDATE, or DELETE statement.
:

expression IN (value1, value2, ... value_n);

sion: The value to test.

value2, ... or value_n: These are the values to test against expression. I
matches expression, then the IN condition will evaluate to true. This is a qu
ny one of the values matches expression.

Y_ID	COUNTRY_NAME	REGION_ID
	United State	1001
	Vietnam	NULL
	Indonesia	NULL
	England	1002
	Germany	1002

```
SELECT * FROM Countries WHERE country_name in ('Engla
```

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C5	Indonesia	NULL
	C2	England	1002
	C3	Germany	1002

SQL IN condition example would return all rows from the countries table w
England, Germany or United State

Comparison Operators: NOT

SQL NOT Condition (also called the NOT Operator) is used to negate a co
T, INSERT, UPDATE, or DELETE statement.

NOT condition

mple:

Y_ID	COUNTRY_NAME	REGION_ID
	United State	1001
	Vietnam	NULL
	Indonesia	NULL
	England	1002
	Germany	1002

```
SELECT * FROM Countries WHERE country_name NOT IN ('Eng
```

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C1	United State	1001
	C4	Vietnam	NULL

'SQL NOT example would return all rows from the country table where the
, Andrew, or Brad.

Comparison Operators: BETWEEN Condition

SQL BETWEEN Condition is used to retrieve values within a range in a SE
T, UPDATE, or DELETE statement.
:

expression BETWEEN value1 AND value2;

sion: A column or calculation.

and value2: These values create an inclusive range that expression is cor

mple:

_ID	COUNTRY_NAME	REGION_ID
	United State	1001
	Vietnam	NULL
	Indonesia	NULL
	England	1002
	Germany	1002
	Brazil	1003
	France	1004

```
SELECT * FROM Countries WHERE region_id BET
```

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C2	England	1002
	C3	Germany	1002
	C6	Brazil	1003

SQL BETWEEN example would return all rows from the contacts table wh
id is between 1002 and 1002 (inclusive)

Comparison Operators: IS NULL Condition

SQL IS NULL Condition is used to test for a NULL value in a SELECT, INS
E, or DELETE statement.

expression IS NULL

sion: The value to test if it is a NULL value.

ok at an example of how to use MySQL IS NULL in a SELECT statement:

TRY_ID	COUNTRY_NAME	REGION_ID
	United State	1001
	Vietnam	NULL
	Indonesia	NULL
	England	1002
	Germany	1002
	Brazil	1003
	France	1004

```
SELECT * FROM Countries WHERE r
```

	COUNTRY_ID	COUNTRY_NAME	RI
▶	C4	Vietnam	NUL
	C5	Indonesia	NUL

SQL IS NULL example will return all records from the contacts table where
s a NULL value.

Comparison Operators: IS NOT NULL

SQL IS NOT NULL condition is used to test for a NOT NULL value in a SE
E, or DELETE statement.

expression IS NOT NULL
sion: The value to test if it is a not NULL value.
e - With SELECT Statement
an example of how to use the MySQL IS NOT NULL condition in a SELEC

_ID	COUNTRY_NAME	REGION_ID
	United State	1001
	Vietnam	NULL
	Indonesia	NULL
	England	1002
	Germany	1002
	Brazil	1003
	France	1004

```
SELECT * FROM Countries WHERE region_id
```

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C1	United State	1001
	C2	England	1002
	C3	Germany	1002
	C6	Brazil	1003
	C7	France	1004

'SQL IS NOT NULL example will return all records from the countries table
id does not contain a null value.

Comparison Operators: LIKE Condition

/SQL LIKE operator is used in a WHERE clause to search for a specified p
n

n: There are two wildcards often used in conjunction with the LIKE operato

Wildcard	Explanation
%	Allows you to match any string of any length (including zero lengt
_	Allows you to match on a single character

Comparison Operators: LIKE Condition

examples of using like operator

E Operator	Description
RE CustomerName LIKE 'a%'	Finds any values that start with "a"
RE CustomerName LIKE '%a'	Finds any values that end with "a"
RE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
RE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
RE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3
RE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

Comparison Operators: LIKE Condition

ple: Display all country names that ends with land

COUNTRY_ID	COUNTRY_NAME	REGION_ID
C1	United State	1001
C4	Vietnam	NULL
C5	Indonesia	NULL
C2	England	1002
C6	Brazil	1003
C3	Netherland	1002
C7	Poland	1004

```
SELECT * FROM Countries WHERE country_name LIKE '%land';
```

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
▶	C2	England	1002
	C3	Netherland	1002
	C7	Poland	1004

Joins

QL joins are used to combine columns from two or more table
mon field between them.

es of Joins:

nner join

Outer join

- Left outer join
- Right outer join
- Full outer join

Cross join

Self join

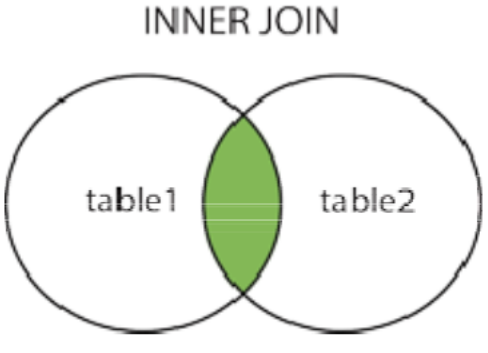
Joins - Types of Joins:

n

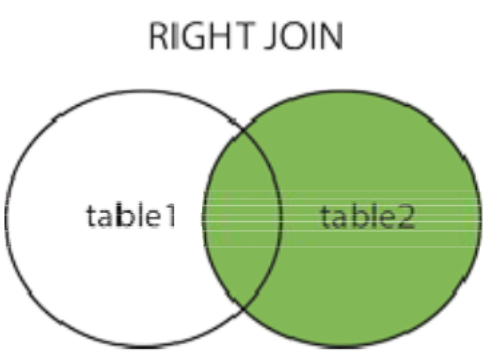
NNER JOIN keyword selects all rows from both tables
ng as there is a match between the columns in both
s

r join

Left outer join: LEFT JOIN keyword returns all rows from
he left table (table1), with the matching rows in the right
able (table2). The result is NULL in the right side when
here is no match.



Right outer join: RIGHT JOIN keyword returns all rows
rom the right table (table2), with the matching rows in
he left table (table1). The result is NULL in the left side
when there is no match



Full outer join : The FULL OUTER JOIN keyword returns
all rows from the left table (table1) and from the right
able (table2).

The FULL OUTER JOIN keyword combines the result of
n LEFT and RIGHT joins

s join

oin

Joins - Sample Table

loyee table

LastName	DepartmentID
rty	31
	33
nberg	33
son	34
	34
	NULL
ms	

Department table

DepartmentID	D
31	Sale
33	Engi
34	Cleri
35	Mark

Joins - Inner Join

table

Name	DepartmentID
	31
	33
	33
	34
	34
	NULL

```
SELECT E.Lastname,E.DepartmentID, D.DepartmentNar
FROM employee E
INNER JOIN department D
ON E.DepartmentID = D.DepartmentID;
```

t table

tmentID	DepartmentName
	Sales
	Engineering
	Clerical
	Marketing

Employee.LastName	Employee.DepartmentID	Departme mentNam
Robinson	34	Clerical
Jones	33	Engineering
Smith	34	Clerical
Heisenberg	33	Engineering
Rafferty	31	Sales

Joins - Left Outer Join

yee table

LastName	DepartmentID
	31
	33
berg	33
on	34
	34
s	NULL

ment table

epartmentID	DepartmentName
	Sales
	Engineering
	Clerical
	Marketing

```
SELECT * FROM employee E
LEFT OUTER JOIN Department D
ON E.DepartmentID = D.DepartmentID;
```

Employee.LastName	Employee.DepartmentID	Depa ment
Jones	33	Engineer
Rafferty	31	Sales
Robinson	34	Clerical
Smith	34	Clerical
Williams	NULL	NULL
Heisenberg	33	Engineer

Joins - Right Outer Join

table	
Name	DepartmentID
	31
	33
	33
	34
	34
	NULL

```
SELECT *
FROM employee
RIGHT OUTER JOIN department
ON employee.DepartmentID = depart
```

t table	
tmentID	DepartmentName
	Sales
	Engineering
	Clerical
	Marketing

Employee.Last Name	Employee.DepartmentID	Department.Dep Name
Smith	34	Clerical
Jones	33	Engineering
Robinson	34	Clerical
Heisenberg	33	Engineering
Rafferty	31	Sales
NULL	NULL	Marketing

Joins - Full Outer Join

table

Name	DepartmentID
	31
	33
	33
	34
	NULL

t table

tmentID	DepartmentName
	Sales
	Engineering
	Clerical
	Marketing

```
#Only work on SQL (use union left a
MySQL)
SELECT * FROM employee
FULL OUTER JOIN department
ON employee.DepartmentID = departm
```

Employee.Last Name	Employee.DepartmentID	Department.Departmt Name
Smith	34	Clerical
Jones	33	Engineering
Robinson	34	Clerical
Williams	NULL	NULL
Heisenberg	33	Engineering
Rafferty	31	Sales
NULL	NULL	Marketing

Joins - CROSS JOIN

s the Cartesian
t of rows from tables
join. In other words, it
oduce rows which
ne each row from the
ble with each row from
cond table

```
CT * FROM employee
S JOIN department;
```

Employee.LastName	Employee.Department ID	Department.D Name
Rafferty	31	Sales
Jones	33	Sales
Heisenberg	33	Sales
Smith	34	Sales
Robinson	34	Sales
Williams	NULL	Sales
Rafferty	31	Engineering
Jones	33	Engineering
Heisenberg	33	Engineering
Smith	34	Engineering
Robinson	34	Engineering
Williams	NULL	Engineering
Rafferty	31	Clerical
Jones	33	Clerical
Heisenberg	33	Clerical
Smith	34	Clerical
Robinson	34	Clerical
Williams	NULL	Clerical
Rafferty	31	Marketing
Jones	33	Marketing
Heisenberg	33	Marketing
Smith	34	Marketing
Robinson	34	Marketing
Williams	NULL	Marketing

Joins - Self Join

Self-join is joining a table to itself

	fullname	date of join	supervisor
	Nick Fury	2008-05-02	NULL
	Tony Stark	2008-05-02	1
	Bruce Banner	2008-06-13	1
	Peter Parker	2017-07-07	2
	Steve Rogers	2012-05-04	1
	Bucky Barnes	2012-05-04	5
	Sam Wilson	2014-04-04	5

```
select A.id,A.fullname,B.id as Sup
from employee2 A, employee2 B
where A.supervisor = B.id;
```

id	fullname	Supervisor_ID	ful
2	Tony Stark	1	Nic
3	Bruce Banner	1	Nic
5	Steve Rogers	1	Nic
4	Peter Parker	2	Tor
6	Bucky Barnes	5	Ste
7	Sam Wilson	5	Ste

Q&A