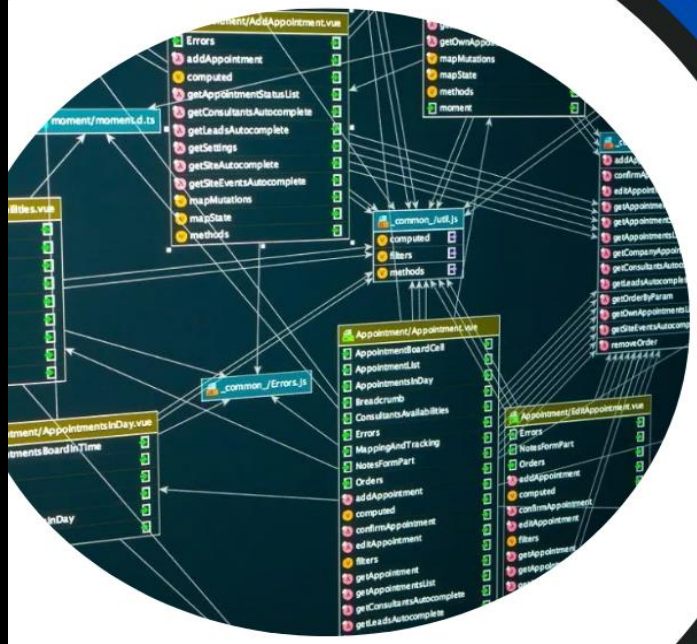
NETCOM TRAINING

NCFE LEVEL 3 CERTIFICATE IN DATA

MySQL Portfolio



2024

PREPARED BY : Anasua Sarkar

Introduction

Welcome to my MySQL portfolio, a collection of projects and exercises completed during my beginner-level course in database management. This portfolio highlights the fundamental concepts of MySQL, including data creation, manipulation, and querying techniques. Throughout these tasks, I've learned how to design and interact with relational databases, gaining hands-on experience that has solidified my understanding of database principles. Whether it's creating tables, writing SQL queries, or optimizing database performance, this portfolio reflects my journey and growth in mastering the essentials of MySQL.

What is a flat file database?

A flat file database is a type of database that stores data in a single table, typically as a plain text file. Each line in the file represents a record, and fields within a record are usually separated by delimiters such as commas (in CSV files), tabs, or spaces. Flat file databases use fieldnames to identify different fields and support basic datatypes such as strings and numbers. They are simple and easy to use for small amounts of data but lack the advanced features and performance capabilities of more complex database systems like relational databases.

Explain the Advantages and Disadvantages of Database Systems

The Advantages of the database system include:

1. **Data Redundancy Control:** The database system minimizes redundancy by consolidating files, ensuring more efficient use of data.
2. **Data Consistency: Reducing** redundancy minimizes inconsistencies and maintains data coherence.
3. **Data Sharing:** The entire organization can access and share the database.
4. **Enhanced data integrity:** Database integrity provides the validity and consistency of stored data. Integrity is usually

expressed in terms of constraints, which are consistency rules that the database is not permitted to violate.

5. **Improved Data Accessibility and Responsiveness**: Improved Data Accessibility and Responsiveness: By integrating the database approach, data can be accessed across departmental boundaries, providing more functionality and better services to users.
6. Improved Maintenance: The database approach provides data independence. Changes in data structure in the database do not affect the application program, simplifying database application maintenance.

The disadvantages of the database system include:

1. **Complexity**: Implementing a database management system can be challenging. It requires training for administrators, designers, and users.
2. ***Size***: A database management system needs a significant amount of main memory and a large disk space to run efficiently. Converting from a file-based system to a database system also involves additional expenses for hardware and training.
3. **Cost**: A multi-user DBMS can be expensive both initially and in terms of annual maintenance costs.
4. **Performance**: Some applications may not run as fast as they did before when using a database approach.
5. **Vulnerability:** The database approach can increase system vulnerability due to centralization. If any component of the database system fails, it can halt operations and affect services for customers.

Data types of SQL

A data type is an attribute that specifies the type of data that the object can hold: integer data, character data, monetary data, date and time data, binary strings, and so on. (1)

Type	Value
CHAR()	A FIXED length string (can contain letters, numbers, and special characters). The size parameter() specifies the column length in characters - can be from 0 to 255. Default is 1. E.g. CHAR(10) will allow 10 characters.
VARCHAR()	A string of variable length up to 255 characters long.
INT	An integer range is from - 2147483648 to 2147483647
DECIMAL	A floating point number that can specify the number permissible digits. E.g. (5,2) permits – 999.99 to 999.99.
DATE	A date in the YY-MM-DD format.
TIME	A time in the HH:MM:SS format.
DATETIME	Combination of the date and time format. Date first and then time. YY-MM-DD HH:MM:SS.
BOOL	Equal to Boolean.

Field modifiers in SQL

Modifiers can be used to control column content.

Type	value
NOT NULL	Insists that no data can be blank – must have a data value in the column.
UNIQUE	Ensure that records may not duplicate any entry in this column.
AUTO_INCREMENT	On numeric columns that automatically generates a number one more than the previous numbers in the column
DEFAULT	Specifies a value to be used where no value is entered for the column when inserting records
PRIMARY KEY	Specifies which column will be used as the primary key for that table.

What is a primary key and why should all tables have a unique identifier?

The primary key is a field which uniquely identifies each record in a table in a relational database. Examples: Employee Number in a employee table, product id in a product table or a auto generated purchase id in a purchase table.

It ensures that each row can be uniquely identified, which is crucial for maintaining data integrity, enabling efficient data retrieval, and establishing relationships between tables. All tables should have a primary key to avoid duplicate records, facilitate indexing, and support reliable data manipulation.

Primary key is product id for this table which is able to identify each row uniquely

Amazon product table

Product ID	Name/Desc	Wholesale Price	List Price	Quantity in Stock
1	Dobble Card game	4.00	5.99	352
2	Monoloply Game	8.99	12.99	241
3	Nerf Elite 2.0 Commander RD-Blaster	3.00	6.66	147
4	Jenga	5.75	10.99	101
5	Crayola Super Tips Washable Markers	1.00	2.85	587

Image: Netcom training

What is a relational database and the advantages of it?

A relational database (RDB) is a way of structuring information in tables, rows, and columns. An RDB has the ability to establish links—or relationships–between information by joining tables, which makes it easy to understand and gain insights about the relationship between various data points.

Benefits of relational databases

The main benefit of the relational database model is that it provides an intuitive way to represent data and allows easy access to related data points. As a result, relational databases are most commonly used by organizations that need to manage large amounts of structured data, from tracking inventory to processing transactional data to application logging.

There are many other advantages to using relational databases to manage and store your data, including:

Flexibility

It’s easy to add, update, or delete tables, relationships, and make other changes to data whenever you need without changing the overall database structure or impacting existing applications.

ACID compliance

Relational databases support ACID (Atomicity, Consistency, Isolation, Durability) performance to ensure data validity regardless of errors, failures, or other potential mishaps.

Ease of use

It's easy to run complex queries using SQL, which enables even non-technical users to learn how to interact with the database.

Collaboration

Multiple people can operate and access data simultaneously. Built-in locking prevents simultaneous access to data when it's being updated.

Built-in security

Role-based security ensures data access is limited to specific users.

Database normalization

Relational databases employ a design technique known as normalization that reduces data redundancy and improves data integrity.

(2)

Image Mongodb

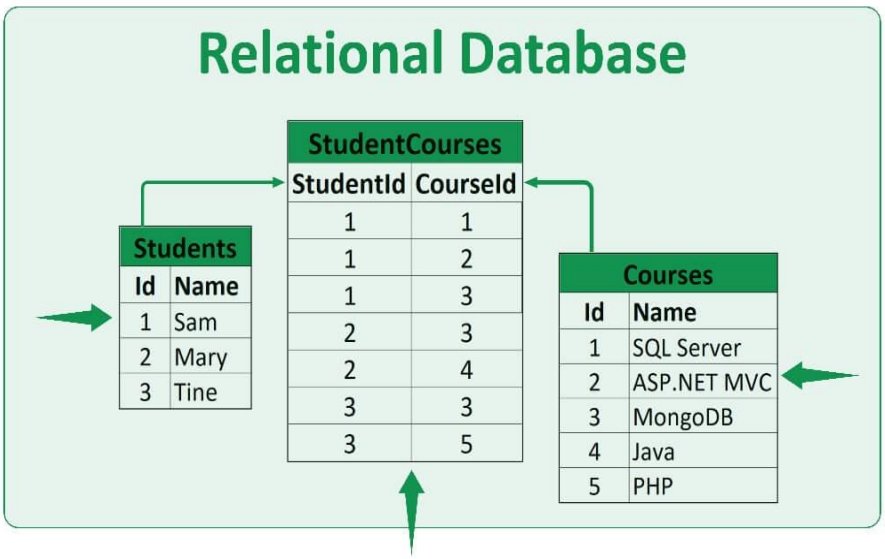
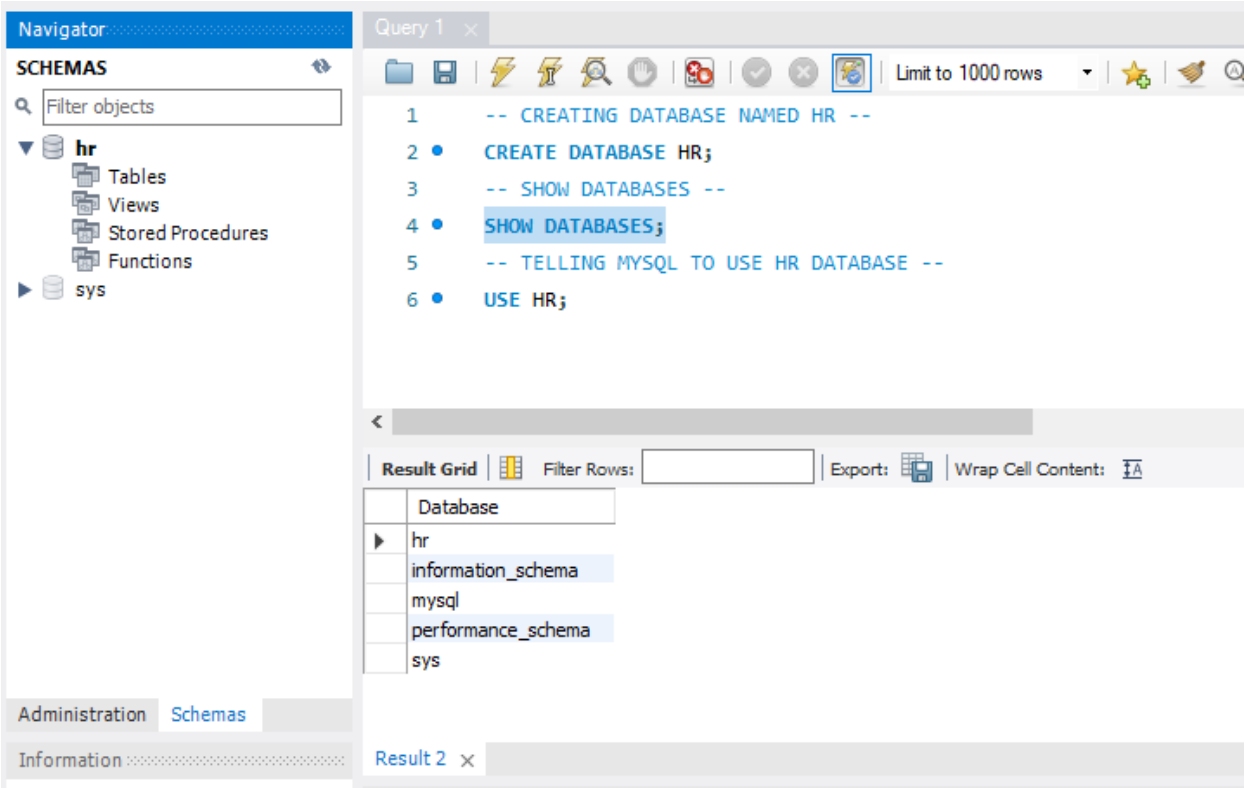


Diagram: Relational database of a training institute

MySQL

Creating a database named HR, Showing it in result grid and using it. We can see that it is in use as it is highlited in Schemas.



Commands are:

```
-- CREATING DATABASE NAMED HR --
CREATE DATABASE HR;
-- SHOW DATABASES --
SHOW DATABASES;
-- TELLING MYSQL TO USE HR DATABASE --
USE HR;
```

Creating a table in Database

```
7     -- CREATING Table NAMED Employee_details --
8  CREATE TABLE employee_details (
9      employee_id INT NOT NULL,
10     job_title VARCHAR(30) NOT NULL,
11     employee_firstname VARCHAR(30) NOT NULL,
12     employee_lastname VARCHAR(30) NOT NULL,
13     contact_no VARCHAR(14) NOT NULL,
14     dob DATE NOT NULL,
15     PRIMARY KEY (employee_id)
16 );
```

Using Explain command us see the table stucture

```
16 );
17 EXPLAIN employee_details;
18
```

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

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
employee_firstname	varchar(50)	NO		NULL	
employee_lastname	varchar(50)	NO		NULL	
dob	date	NO		NULL	
job_title	varchar(50)	NO		NULL	
phone_no	varchar(14)	NO		NULL	

Result 2

Output

```
-- Using DESCRIBE or EXPLAIN command
--  to see the table stucture --
EXPLAIN employee_details;
DESCRIBE employee_details;
```

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

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
employee_firstname	varchar(50)	NO		NULL	
employee_lastname	varchar(50)	NO		NULL	
dob	date	NO		NULL	
age	int	YES		NULL	
job_title	varchar(50)	NO		NULL	
phone_no	varchar(14)	NO		NULL	
salary	decimal(8,2)	NO		NULL	

Result 1

Inserting Data Into the table

```
INSERT INTO employee_details(employee_id,employee_firstname,
employee_lastname,dob,job_title,phone_no,email_id)
VALUES
(1,"Jhon","Doe","1989-08-07","Mnager","072345679","abc@email.com"),
(2,"Mark","Cross", "1990-02-04","Accountant","072345679","ehg@email.com"),
(3,"Ana","Simon", "1999-06-07","IT","072345679","abertc@email.com"),
(4,"Peter","Jackson", "1989-07-07","IT","072345679","ahgc@email.com");
```

-- Inserting Data into Table Adding Two More Records --

```
INSERT INTO employee_details(employee_id,employee_firstname,employee_lastname,
dob,job_title,phone_no,email_id)
VALUES
(5,"Aniya","Smith","1989-08-07","Mnager","07256767459","aniyac@email.com"),
(6,"Charli","Jones", "1976-03-04","Accountant","0734565679","charli@email.com");
```

Reading all the data from the table

```
-- Reading all the data from the table--
SELECT * FROM  employee_details;
```

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

employee_id	employee_firstname	employee_lastname	dob	job_title	phone_no	email_id
1	Jhon	Doe	1989-08-07	Manager	072345679	jhondoe@email.com
2	Mark	Cross	1990-02-04	Accountant	072345679	ehg@email.com
3	Ana	Simon	1999-06-07	IT	072345679	abertc@email.com
4	Peter	Jackson	1989-07-07	IT	0774455666	ahgc@email.com
5	Aniya	Smith	1989-08-07	Manager	07256767459	aniyac@email.com
6	Charli	Jones	1976-03-04	Accountant	0734565679	charli@email.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Updating table

--Updating record of employee name "Jhon". Changing his phone number to 0774455678.--

```
-- updating a record in the table--
UPDATE employee_details
SET  phone_no = '0774455678'
WHERE employee_firstname = 'Jhon';
-- Reding the updated record --
SELECT * FROM employee_details WHERE employee_firstname = 'Jhon';
```

-- Reding the updated record –

SELECT * FROM employee_details WHERE employee_firstname = 'Jhon';

```
UPDATE employee_details
SET  phone_no = '0774455678'
WHERE employee_firstname = 'Jhon';
SELECT * FROM employee_details WHERE employee_firstname = 'Jhon';
```

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

employee_id	employee_firstname	employee_lastname	dob	job_title	phone_no	salary	email_id
1	Jhon	Doe	1989-08-07	Manager	0774455678	27000.00	jhondoe@email.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Altering a table adding a new column "salary"

```
-- Adding a column salary to employee_details  after column phoneno --
ALTER TABLE employee_details
ADD COLUMN salary INT AFTER phone_no;
SELECT * FROM employee_details;
```

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

employee_id	employee_firstname	employee_lastname	dob	job_title	phone_no	salary	email_id
1	Jhon	Doe	1989-08-07	Manager	0774455678	NULL	jhondoe@email.com
2	Mark	Cross	1990-02-04	Accountant	072345679	NULL	ehg@email.com
3	Ana	Simon	1999-06-07	IT	072345679	NULL	abertc@email.com
4	Peter	Jackson	1989-07-07	IT	0774455666	NULL	ahgc@email.com
5	Aniya	Smith	1989-08-07	Manager	07256767459	NULL	aniyac@email.com
6	Charli	Jones	1976-03-04	Accountant	0734565679	NULL	charli@email.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Adding data to the newly add column salary

```
-- Adding Values to salary column where job title is Accountant --
UPDATE employee_details
SET salary = 25000
WHERE job_title = "Accountant";
SELECT employee_firstname, job_title, salary
```

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

employee_firstname	job_title	salary
Jhon	Manager	NULL
Mark	Accountant	25000
Ana	IT	NULL
Peter	IT	NULL
Aniya	Manager	NULL
Charli	Accountant	25000

Adding data to the newly add column salary using logical operator " IN"

```
-- Adding Values to salary column where employee_id is 1 and 5 --
UPDATE employee_details
SET salary = 28000
WHERE employee_id IN (1 , 5);
SELECT * FROM employee_details;
```

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

employee_id	employee_firstname	employee_lastname	dob	job_title	phone_no	salary	email_id
1	Jhon	Doe	1989-08-07	Manager	0774455678	28000	jhondoe@email.com
2	Mark	Cross	1990-02-04	Accountant	072345679	25000	ehg@email.com
3	Ana	Simon	1999-06-07	IT	072345679	NULL	abertc@email.com
4	Peter	Jackson	1989-07-07	IT	0774455666	NULL	ahgc@email.com
5	Aniya	Smith	1989-08-07	Manager	07256767459	28000	aniyac@email.com
6	Charli	Jones	1976-03-04	Accountant	0734565679	25000	charli@email.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Adding Values to salary column where employee first name is Ana or Peter using logical operator "OR"

```
-- Adding Values to salary column using logical operetor OR --
UPDATE employee_details
SET salary = 26000
WHERE employee_firstname ="Ana" OR  employee_firstname="Peter";
SELECT * FROM employee_details;
```

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

employee_id	employee_firstname	employee_lastname	dob	job_title	phone_no	salary	email_id
1	Jhon	Doe	1989-08-07	Manager	0774455678	28000	jhondoe@email.com
2	Mark	Cross	1990-02-04	Accountant	072345679	25000	ehg@email.com
3	Ana	Simon	1999-06-07	IT	072345679	26000	abertc@email.com
4	Peter	Jackson	1989-07-07	IT	0774455666	26000	ahgc@email.com
5	Aniya	Smith	1989-08-07	Manager	07256767459	28000	aniyac@email.com
6	Charli	Jones	1976-03-04	Accountant	0734565679	25000	charli@email.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Altering datatype of salary column to DECIMAL from INT

```
-- Altering datatype of salary column to DECIMAL  from INT --
ALTER TABLE employee_details
MODIFY salary DECIMAL (8,2) NOT NULL;
DESCRIBE employee_details;
```

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

Field	Type	Null	Key	Default	Extra
employee_id	int	NO	PRI	NULL	
employee_firstname	varchar(50)	NO		NULL	
employee_lastname	varchar(50)	NO		NULL	
dob	date	NO		NULL	
job_title	varchar(50)	NO		NULL	
phone_no	varchar(14)	NO		NULL	
salary	decimal(8,2)	NO		NULL	
email_id	varchar(50)	NO		NULL	

Finding Minimum Salary Using min() Function

```
SELECT MIN(salary) FROM employee_details;
```

MIN() FUNCTION

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

MIN(salary)
25000.00

Finding Maximum Salary Using max() Function

```
78 •  SELECT MAX(salary) FROM employee_details;
```

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

	MAX(salary)
▶	28000.00

MAX() FUNCTION

Finding Average Salary Using AVG() And COUNT()Function

```
81 •  SELECT AVG(salary) FROM employee_details;
```

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

	AVG(salary)
▶	26333.333333

AVG() FUNCTION

```
83 •  SELECT COUNT(salary) FROM employee_details;
```

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

	COUNT(salary)
▶	6

COUNT() FUNCTION

Finding specific column and sorting in descending order on the values of"dob" column using order by and DESC

```
SELECT
    employee_id, employee_firstname,
    employee_lastname, dob
FROM
    employee_details
ORDER BY dob DESC;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

employee_id	employee_firstname	employee_lastname	dob
3	Ana	Simon	1999-06-07
2	Mark	Cross	1990-02-04
1	Jhon	Doe	1989-08-07
5	Aniya	Smith	1989-08-07
4	Peter	Jackson	1989-07-07
6	Charli	Jones	1976-03-04

Sorting using ASC

```
SELECT
    employee_id, employee_firstname,
    job_title, salary
FROM
    employee_details
ORDER BY salary ASC;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

employee_id	employee_firstname	job_title	salary
2	Mark	Accountant	25000.00
6	Charli	Accountant	25000.00
3	Ana	IT	26000.00
4	Peter	IT	26000.00
1	Jhon	Manager	28000.00
5	Aniya	Manager	28000.00

Using" WHERE "Clause to Specify A Condition & Using "BETWEEN" and "AND" To Specify A Range

```
SELECT employee_firstname, salary
FROM employee_details
WHERE salary BETWEEN 20000 AND 25000;
```

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

employee_firstname	salary
Mark	25000.00
Charli	25000.00

Using ">"to find salary greater than 25000 and sorting it using DESE

```
SELECT employee_firstname, salary , Phone_no
FROM employee_details
WHERE salary> 25000
ORDER BY salary DESC;
```

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

employee_firstname	salary	Phone_no
Jhon	28000.00	0774455678
Aniya	28000.00	07256767459
Ana	26000.00	072345679
Peter	26000.00	0774455666

Using "GROUP BY" To Identify Different Group Of Salary and Finding How Many Employees Are There In Each Group using (COUNT*)

```
SELECT salary, count(*)
FROM employee_details
WHERE salary> 25000
GROUP BY salary
ORDER BY salary ASC;
```

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

salary	count(*)
26000.00	2
28000.00	2

Showing "dob"column as "birth_day" using AS

```
-- selecting date of birth --
SELECT dob AS birth_day FROM employee_details ;
```

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

birth_day
1989-08-07
1990-02-04
1999-06-07
1989-07-07
1989-08-07
1976-03-04

Using TIMESTAMPDIFF() function to calculate age of employees

The MySQL **TIMESTAMPDIFF()** function is used to calculate the difference between two datetime or, date expressions.

This function accepts two datetime or date expressions as parameter values, calculates the difference between them and returns the result. One of the arguments can be date and the other a datetime expression.

(3)

```
25      -- Age claculation --
26 ●    SELECT employee_firstname,employee_lastname,dob,
27             TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS age
28      FROM  employee_details;
29
```

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

	employee_firstname	employee_lastname	dob	age
▶	Jhon	Doe	1989-08-07	34
	Mark	Cross	1990-02-04	34
	Ana	Simon	1999-06-07	25
	Peter	Jackson	1989-07-07	35
	Aniya	Smith	1989-08-07	34
	Charli	Jones	1976-03-04	48

Adding a new column "age" to the table and auto calculating age from "dob"

```
-- adding a new column age to the table and auto calculating age from dob -
ALTER TABLE employee_details
ADD COLUMN age INT  AFTER dob;
--  Autocalculating Age  --
UPDATE employee_details SET age = TIMESTAMPDIFF(YEAR, dob, CURDATE());
select * from employee_details;
```

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

employee_id	employee_firstname	employee_lastname	dob	age	job_title	phone_no
1	Jhon	Doe	1989-08-07	34	Manager	0774455678
2	Mark	Cross	1990-02-04	34	Accountant	072345679
3	Ana	Simon	1999-06-07	25	IT	072345679
4	Peter	Jackson	1989-07-07	35	IT	0774455666
5	Aniya	Smith	1989-08-07	34	Manager	07256767459
6	Charli	Jones	1976-03-04	48	Accountant	0734565679
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Creating a new database

-- Creating a new database --

-- Use the newly created database –

```
-- Creating a new database --
CREATE DATABASE patient;
-- Using the newly created database --
USE patient;
-- Creating a table in database with primary key --
```

Creating a table in database with primary key

-- Creating a table in database with primary key –

```sql
-- Creating a table in database with primary key --
CREATE TABLE patient_detail(
    patient_id INT NOT NULL,
    patient_firstname VARCHAR(30) NOT NULL,
    patient_lastname VARCHAR(30) NOT NULL,
    dob DATE NOT NULL,
    gender boolean NOT NULL,
    phone_no VARCHAR(14) NOT NULL,
    primary key(patient_id)
);
```

Explain table detail

```sql
EXPLAIN patient_detail;
ALTER TABLE patient_detail
```

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

Field	Type	Null	Key	Default	Extra
patient_id	int	NO	PRI	NULL	auto_increment
patient_firstname	varchar(30)	NO		NULL	
patient_lastname	varchar(30)	NO		NULL	
dob	date	NO		NULL	
gender	char(1)	NO		NULL	
phone_no	varchar(14)	NO		NULL	

What methods could we use to collect data and build ourselves a dataset?

Please briefly explain what each point is. Give an example of the method where possible and talk about how it could be advantageous.

● Surveys, questionnaires and quizzes :

Surveys are a popular method for collecting data from people. They can be conducted in person, over the phone, through mail, or online, with structured or unstructured questions. These methods are helpful for gathering information.

Advantages of using this approach include:

- Identifying customer preferences
- Evaluating satisfaction levels
- Assessing any changes
- Monitoring changes in satisfaction levels

• **Interviews and focus groups:** Interviews and focus groups are two different methods of collecting information. Interviews involve one-on-one conversations with individuals They can be structured or unstructured, with open-ended or closed-ended questions.

On the other hand, a focus group is a qualitative research technique that involves a small group of five to 10 people discussing a specific topic or issue. A moderator leads the group, asks open-ended questions, and facilitates participant discussion and interaction.

Advantages:

- Both methods provide rich data and the opportunity for moderators to ask follow-up questions.
- Focus groups are relatively quick and cost-effective and allow for information to be gathered in the participants' own words.
- Information can be obtained more quickly than if individuals were interviewed separately.

Observation : Observation involves watching and recording the behavior of people or objects, commonly used in anthropology, psychology, and sociology.

Advantages:

- Accuracy : Data collected through this method is more accurate compared to other methods because there is no intervention and no pressure on people to influence their behavior.
- Easy to organize: In a natural environment, no need to "organize" anything. No need to find participants or rent an office. Saves money and can revisit the same setting to repeat the process.
- Flexibility: You can skip scheduled observations, change location, and subjects at any time. Adjust your approach to observe different demographics in crowded areas.

Experiment: Experiments involve manipulating one or more variables to observe the effect on another variable. This method is commonly used in scientific research to establish causal relationships between variables. Experiments can be conducted in controlled environments (laboratories) or natural settings (field experiments).

Example: A pharmaceutical company conducting clinical trials to test the efficacy of a new drug. (3)

Web scraping : Web scraping involves extracting data from websites using automated software tools. This method is often used in fields such as data science and machine learning

You can gather information faster than traditional data extraction methods thanks to web scraping tools. Compiling the data by hand could take days or months, but scraping would extract hundreds or thousands of data within a few minutes or hours

Sensors: Sensors are used to collect data from physical objects or environments. This method is often used in fields such as engineering and environmental science.

Advantages:

- Improving Efficiency: Monitoring electricity usage in real-time helps companies reduce operating costs.
- Improving Safety: Constant monitoring reduces workplace accidents.
- Predictive Maintenance: Predictive maintenance helps in repairing machinery before it fails; the machines get repaired before they fail.

Crowdsourcing*:* Crowdsourcing involves obtaining work, information, or opinions from a large group of people who submit their data via the Internet, social media, and smartphone apps. (4)

Advantages:

Cost-effective: the company only pays for bugs which are found instead of an hourly or salaried
• rate which professional testers would receive Vast range of users provide huge diversity in their experiences
• Allows for testing with all kinds of different parameters, such as with different connection
• speeds, browsers, and devices to which the core testing team may not have access Larger group is more likely to find reproducible bugs than a handful of testers
• Lack of bias towards the company can be expected of testers
• Monetary value of both of the above= more thorough testing for an equal price range in a
• shorter amount of time sans contract and overhead (5)

Scenario

Your Company has asked you to obtain a dataset about Internet usage in the UK broken down by different geographical regions between 2014 and 2020. The data should show numbers of recent and lapsed internet users. Your company does not want to spend any money on this data.

Preparing a dataset for migration

TABLE 1A: RECENT AND LAPSED INTERNET USERS AND INTERNET NON-USERS, UK, 2013 TO 2020

Persons aged 16 years and over

	Used in the last 3 months									
	2013	2014	2015	2016	2017	2018	2019	2020	2013	2014
All adults	**42,243**	**43,457**	**44,671**	**45,917**	**46,742**	**47,560**	**48,130**	**49,041**	**1,275**	**1,142**
Age group (years)										
16-24	7,075	7,074	7,155	7,129	7,036	6,992	6,877	6,844	55	23
25-34	8,457	8,660	8,582	8,720	8,815	8,894	8,895	8,908	93	51
35-44	7,952	7,900	8,053	8,129	8,118	8,145	8,243	8,339	109	89
45-54	8,005	8,290	8,498	8,686	8,803	8,814	8,810	8,716	212	187
55-64	5,821	6,060	6,361	6,607	6,888	7,189	7,495	7,796	250	241
65-74	3,562	3,939	4,390	4,721	5,031	5,264	5,339	5,504	315	306
75+	1,371	1,534	1,632	1,925	2,050	2,262	2,471	2,933	242	245

Step1: copy the datset to a excel file.

Step 2:change cells data format from general to number.

Step 3:save the file in **.CSV** format

M7

	A	B	C	D	E	F	G	H	I	J
1	Age group	2013	2014	2015	2016	2017	2018	2019	2020	
2	16-24	7075	7074	7155	7129	7036	6992	6877	6844	
3	25-34	8457	8660	8582	8720	8815	8894	8895	8908	
4	35-44	7952	7900	8053	8129	8118	8145	8243	8339	
5	45-54	8005	8290	8498	8686	8803	8814	8810	8716	
6	55-64	5821	6060	6361	6607	6888	7189	7495	7796	
7	65-74	3562	3939	4390	4721	5031	5264	5339	5504	
8	75++	1371	1534	1632	1925	2050	2262	2471	2933	
9										

Now the datset is ready to import in MySql.

Importing data from .CSV File format data set to MySql

Creating new database for data import

```
1 •  CREATE DATABASE internet_usage;
```

Database using

```
3 •  USE internet_usage;
```

Then using import data wizard, we import data into the ”internet_user” table.

CHANGE COLUMN `Age group (years)` To Age_group_years

- ```
  ALTER TABLE internet_user
  CHANGE COLUMN `Age group (years)` Age_group_years VARCHAR(30);
  ```

Deleting row where Age_group_years = 'Age group (years)';

```
DELETE FROM internet_user
WHERE Age_group_years = 'Age group (years)';
```

Adding a primary key

```
ALTER TABLE internet_user
MODIFY COLUMN Age_group_years  VARCHAR(30) PRIMARY KEY FIRST;
```

Table internate_user after all modifications

Result Grid | Filter Rows: | Edit: | Export/Import:

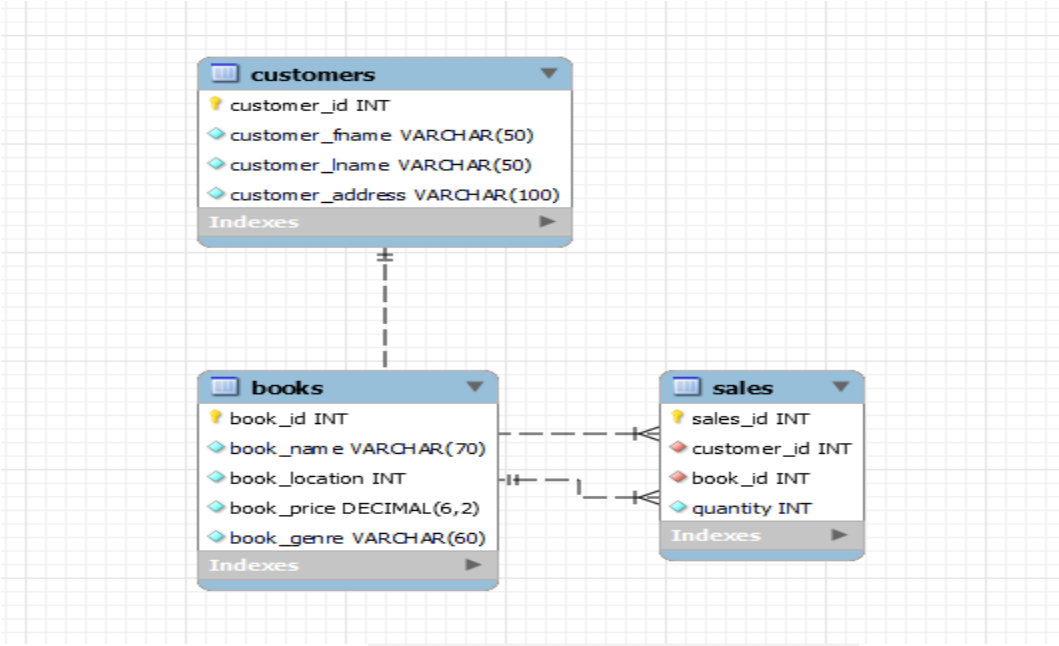
Age_group_years	2013	2014	2015	2016	2017	2018	2019	2020
16-24	7075	7074	7155	7129	7036	6992	6877	6844
25-34	8457	8660	8582	8720	8815	8894	8895	8908
35-44	7952	7900	8053	8129	8118	8145	8243	8339
45-54	8005	8290	8498	8686	8803	8814	8810	8716
55-64	5821	6060	6361	6607	6888	7189	7495	7796
65-74	3562	3939	4390	4721	5031	5264	5339	5504
75++	1371	1534	1632	1925	2050	2262	2471	2933
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
```

TASK:

1) Create a brand new database
2) Create 2 flat file tables (keep them simple and ensure theres numerical fields in there for ease)
3) For points 4 & 5 below, Please ensure to use the Auto_Increment function
4) First table should contain 8 records in total. Please insert 7 records manually and the 8th record must be added to the table AFTER the table has been created
5) Second table should contain 5 records in total. Please insert 6 records manually and then delete the 6th record AFTER the table has been created (remember the drop function)
6) Create a 3rd relational table to link your first 2 tables together. There must be 10 records in this table.
7) Run some simple queries on the last table to extract any information of your choice. (minimum 6 queries)
8) Practice complex queries (joins)
9) Write a brief report to go in your portfolio explaining your work and how you worked in a team.

Pointers:

- Try to use as much queries and updates using all the knowledge you have acquired in SQL so far. The more you can use, the better it looks regardless of how simple you may think it is.
- Remember to use Logical and Boolean operators
- Use google and youtube and the PDFs for help



ER Diagram

```sql
-- CREATING DATABASE NAMED book_store --
CREATE DATABASE book_store;
-- SHOW DATABASES --
SHOW DATABASES;
-- TELLING MYSQL TO USE book_store DATABASE --
USE book_store;
```

```sql
-- This Alters the table by modifying the books price to a decimal --
ALTER TABLE books
MODIFY book_price DECIMAL (6,2) NOT NULL;
```

Limit to 1000 rows

```sql
-- create a table called books --
CREATE TABLE books(
    book_id INT NOT NULL AUTO_INCREMENT,
    book_name VARCHAR(70) NOT NULL,
    book_location INT NOT NULL,
    book_price INT NOT NULL,
    book_genre VARCHAR (60) NOT NULL,
    PRIMARY KEY (book_id)
    );
    DESCRIBE books;
```

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

Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	auto_increment
book_name	varchar(70)	NO		NULL	
book_location	int	NO		NULL	
book_price	decimal(6,2)	NO		NULL	
book_genre	varchar(60)	NO		NULL	

```
28   --  Inserting information into a table --
29 • INSERT INTO books (book_name, book_location, book_price, book_genre) VALUES
30   ("Ghost",123, 10.99, "Horror"),
31   ("Junglebook",135, 12.99, "Adventure"),
32   ("LoveisBlind",142, 15.99, "Romance"),
33   ("Titanic",136, 5.99, "Romance"),
34   ("MySQL",706, 10.99, "Education"),
35   ("Flo",756, 0.99, "Hiphop"),
36   ("Commando",926, 10.99, "Action");
37
```

```
41 • INSERT INTO books (book_name, book_location, book_price, book_genre) VALUES
42   ("Python",998, 13.99, "Education");
43
```

```
38   -- See the  records inserted into the table books --
39 • SELECT * FROM books;
```

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

book_id	book_name	book_location	book_price	book_genre
1	Ghost	123	10.99	Horror
2	Junglebook	135	12.99	Adventure
3	LoveisBlind	142	15.99	Romance
4	Titanic	136	5.99	Romance
5	MySQL	706	10.99	Education
6	Flo	756	0.99	Hiphop
7	Commando	926	10.99	Action
8	Python	998	13.99	Education
NULL	NULL	NULL	NULL	NULL

```sql
CREATE TABLE customers(
    customer_id INT NOT NULL AUTO_INCREMENT,
    customer_fname VARCHAR (50) NOT NULL,
    customer_lname VARCHAR (50) NOT NULL,
    customer_address VARCHAR (100) NOT NULL,
    PRIMARY KEY (customer_id)
    );

EXPLAIN customers;
```

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

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	auto_increment
customer_fname	varchar(50)	NO		NULL	
customer_lname	varchar(50)	NO		NULL	
customer_address	varchar(100)	NO		NULL	

```sql
-- Inserting information into our table customers --
INSERT INTO customers(customer_fname, customer_lname, customer_address) VALUES
    ('Ayesha', 'Anwar',  "London"),
    ('Anna',     'Dow',  "Birmingham"),
    ('Niouma',  'Gaku',  "Manchester"),
    ('Ayaan',  'Haque',  "Leeds"),
    ('Elenor', 'Thyme',  "London"),
    ('Mowreen',  'Fay',  "Derby"),
    ('Harman', 'Smith',  "London"),
    ('Jack', 'Jones',    "Manchester");
```

```sql
-- See the  table with data records inserted --
--  into the table structure --
SELECT * FROM customers;
```

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

customer_id	customer_fname	customer_lname	customer_address
1	Ayesha	Anwar	London
2	Anna	Dow	Birmingham
3	Niouma	Gaku	Manchester
4	Ayaan	Haque	Leeds
5	Elenor	Thyme	London
6	Mowreen	Fay	Derby
7	Harman	Smith	London
8	Jack	Jones	Manchester
NULL	NULL	NULL	NULL

```sql
-- Count the number of customer in each city --
SELECT  customer_address, COUNT(*)
FROM customers
GROUP BY customer_address;
```

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

customer_address	COUNT(*)
London	3
Birmingham	1
Manchester	2
Leeds	1
Derby	1

```sql
-- Creating Table Named Sales --
CREATE TABLE sales(
sales_id INT AUTO_INCREMENT NOT NULL,
customer_id INT NOT NULL,
book_id  INT NOT NULL,
quantity INT NOT NULL,
FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
FOREIGN KEY (book_id) REFERENCES books(book_id),
PRIMARY KEY (sales_id)
);
```

```sql
-- Inserting Values Into sales table.--
INSERT INTO sales(customer_id,book_id,quantity) VALUES
(6, 2, 7),
(2, 2, 6),
(1, 5, 4),
(1, 3, 7),
(1, 1, 8),
(2, 5, 7),
(2, 3, 3);
```

```
-- See the table with data records inserted --
-- into the table structure --

SELECT * FROM sales;
```

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

sales_id	customer_id	book_id	quantity
1	3	2	2
2	1	4	3
3	2	5	1
4	6	3	2
5	5	1	1
6	6	2	7
7	2	2	6
8	1	5	4
9	1	3	7
10	1	1	8
11	2	5	7
12	2	3	3
NULL	NULL	NULL	NULL

```sql
-- Finding distinct customer address --
SELECT DISTINCT customer_address
FROM customers;
```

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

customer_address
London
Birmingham
Manchester
Leeds
Derby

```sql
-- Using JOIN with ORDER BY Clause --
SELECT b.book_id, b.book_name, b.book_location,
b.book_price, b.book_genre, c.customer_id, c.customer_address
FROM sales s
JOIN books b ON b.book_id = s.book_id
JOIN customers c ON c.customer_id = s.customer_id
ORDER BY customer_id ASC;
```

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

book_id	book_name	book_location	book_price	book_genre	customer_id	customer_address
4	Titanic	136	5.99	Romance	1	London
5	MySQL	706	10.99	Education	1	London
3	LoveisBlind	142	15.99	Romance	1	London
1	Ghost	123	10.99	Horror	1	London
5	MySQL	706	10.99	Education	2	Birmingham
2	Junglebook	135	12.99	Adventure	2	Birmingham
5	MySQL	706	10.99	Education	2	Birmingham
3	LoveisBlind	142	15.99	Romance	2	Birmingham
2	Junglebook	135	12.99	Adventure	3	Manchester
1	Ghost	123	10.99	Horror	5	London
3	LoveisBlind	142	15.99	Romance	6	Derby
2	Junglebook	135	12.99	Adventure	6	Derby

```sql
-- Using alias name to a field and using GROUP BY  --
SELECT  customer_address, COUNT(*) AS Number_Of_Customers
FROM customers
GROUP BY customer_address;
```

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

customer_address	Number_Of_Customers
London	3
Birmingham	1
Manchester	2
Leeds	1
Derby	1

TASK:

1) *Convert Excel files "Customers" and "Orders" to CSV (Remember, when importing to workbench, if you encounter any decoding error, you can convert the CSV file to a JSON file and import the JSON file. This should solve the issue). Please find link below in "pointers" section to convert CSV to JSON file.*
2) *Create a brand new database*
3) *Upload the Customers and Orders CSV (or JSON) files to your SQL Workbench or database.*
4) *View both table structure and contents of each table.*
5) *Apply the joins below using both tables*
    - *Simple*
    - *Inner*
    - *Left*
    - *Right*
    - *Full*
6) *Calculate the total order amount for each customer. (Think about everything we have covered so far. This is a challenging one. You will need google to help you. DO NOT USE CHAT GPT)*
7) *Apply appropriate filter to show customers who have placed orders with an amount greater than $1000.*
8) *List all orders by each customer in descending order of order amount.*
9) *Join with date filtering: Select orders placed in January 2023. (This is again to challenge you. It requires you to apply a combination of things we covered in class: Joins and filters)*
10) *Write a brief report to go in your portfolio explaining your work and how you worked in a team.*
11) *Take screenshots of all your work and put in your portfolios with appropriate comments.*

Pointers:

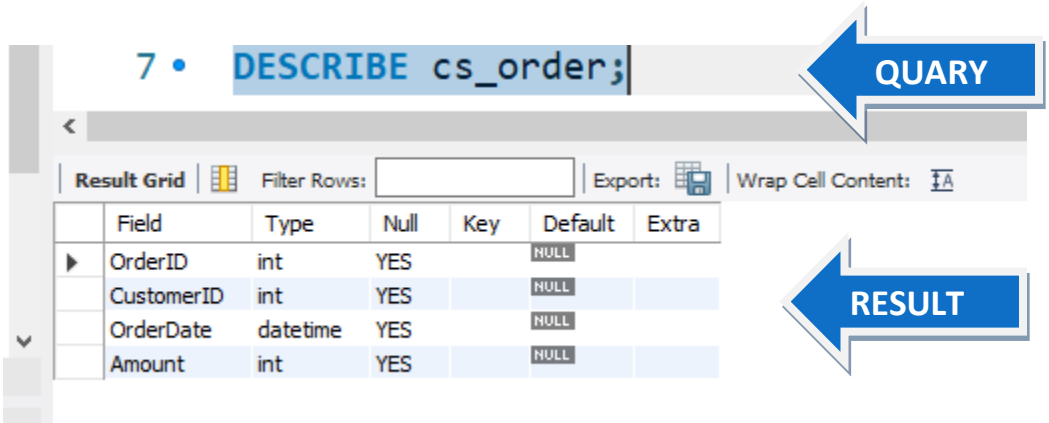
- *Use google and youtube and the PDFs for help*
- *Please DO NOT use chat gpt.*
- *If you can solve all the above, you are then able to use types of joins, aggregate functions, filtering, and ordering in SQL.*
- *Link to convert CSV file to JSON file format IF REQUIRED: https://tableconvert.com/csv-to-json*

TASK 1, 2 AND 3

```
-- CREATING DATABASE NAMED customer_order--
CREATE DATABASE customer_order;
-- SHOW DATABASES --
SHOW DATABASES;
-- TELLING MYSQL TO USE HR DATABASE --
USE customer_order;
```

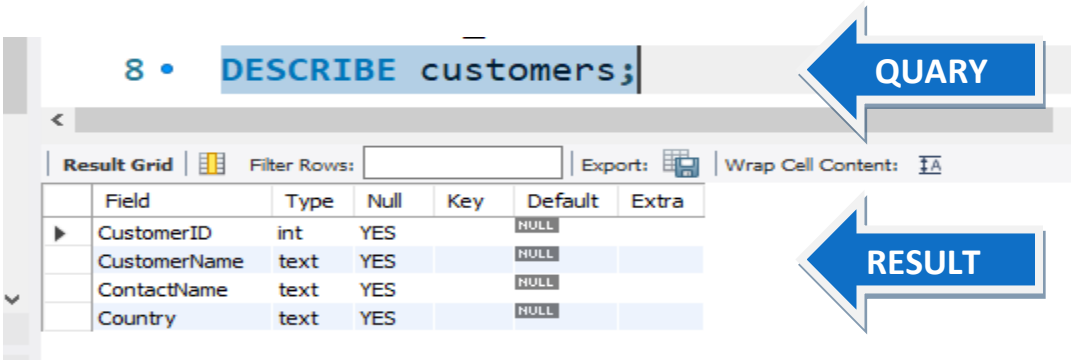
Task 4: View both table structure and contents of each table.

Structure of table cs_order;



Field	Type	Null	Key	Default	Extra
OrderID	int	YES		NULL	
CustomerID	int	YES		NULL	
OrderDate	datetime	YES		NULL	
Amount	int	YES		NULL	

Structure of table customer



Field	Type	Null	Key	Default	Extra
CustomerID	int	YES		NULL	
CustomerName	text	YES		NULL	
ContactName	text	YES		NULL	
Country	text	YES		NULL	

*-

Contain of tablecs_order

```
13 •  SELECT * FROM cs_order;
14
```

QUARY

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

OrderID	CustomerID	OrderDate	Amount
10255	7	2023-01-08 00:00:00	500
10256	8	2023-01-09 00:00:00	1400
10257	9	2023-01-10 00:00:00	900
10258	10	2023-01-11 00:00:00	2000
10259	11	2023-01-12 00:00:00	1000
10260	12	2023-01-13 00:00:00	1500
10261	13	2023-01-14 00:00:00	750
10262	14	2023-01-15 00:00:00	1350
10263	15	2023-01-16 00:00:00	1100
10264	16	2023-01-17 00:00:00	950
10265	17	2023-01-18 00:00:00	500
10266	18	2023-01-19 00:00:00	1250

RESULT

Content of table customer

```
15 •  SELECT * FROM customers;
16
```

QUARY

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

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico
4	Around the Horn	Thomas Hardy	UK
5	Berglunds snabbköp	Christina Berglund	Sweden
6	Blauer See Delikatessen	Hanna Moos	Germany
7	Blondel père et fils	Frédérique Citeaux	France
8	Bólido Comidas preparadas	Martín Sommer	Spain
9	Bon app'	Laurence Lebihan	France
10	Bottom-Dollar Markets	Elizabeth Lincoln	Canada
11	Cactus Comidas para llevar	Patricio Simpson	Argentina
12	Centro comercial Moctezuma	Francisco Chang	Mexico
13	Chop-suey Chinese	Yang Wang	Switzerl...
14	Comércio Mineiro	Pedro Afonso	Brazil
15	Consolidated Holdings	Elizabeth Brown	UK
16	Dahlia Flowers Shop	Timothy Carter	USA
17	Eastern Connection	Ann Devon	UK
18	Ernst Handel	Roland Mendel	Austria
19	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Spain
20	Folies gourmandes	Martine Rancé	France

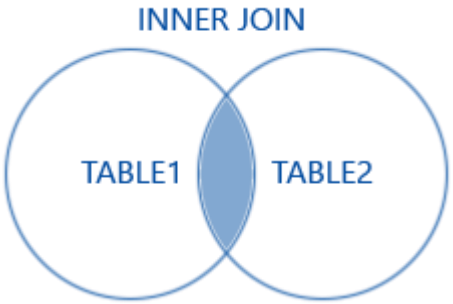
RESULT

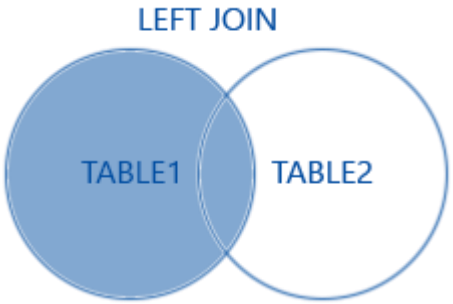
MySQL Joining Tables

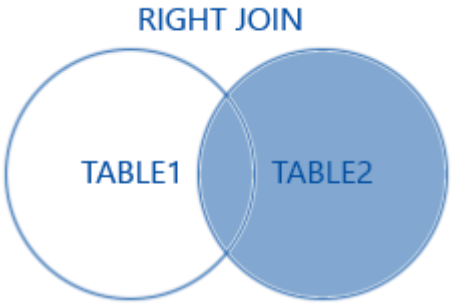
A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Supported Types of Joins in MySQL

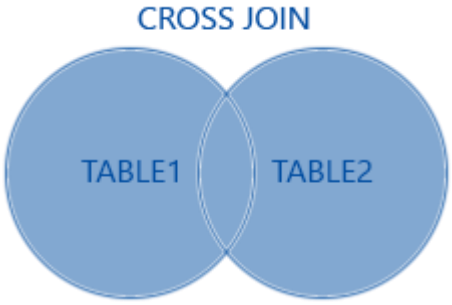
- **INNER JOIN:** Returns records that have matching values in both tables
- **LEFT JOIN*:*** Returns all records from the left table, and the matched records from the right table
- **RIGHT JOIN***:* Returns all records from the right table, and the matched records from the left table
- **CROSS JOIN/FULL JOIN*:*** Returns all records from both tables







CROSS JOIN/FULL JOIN



(7)

Task 5: joins(left ,right, inner, outer and full join)

LEFT JOIN Syntax

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

Note: The LEFT JOIN keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders).

```
30 • SELECT *FROM cs_order
31   LEFT JOIN   customers  ON cs_order.CustomerID = customers.CustomerID;
32
```

QUARY

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

OrderID	CustomerID	OrderDate	Amount	CustomerID	CustomerName	ContactName	Country
10248	1	2023-01-01 00:00:00	440	1	Alfreds Futterkiste	Maria Anders	Germany
10249	2	2023-01-02 00:00:00	1866	2	Ana Trujillo Emparedados y helados	Ana Trujillo	[illegible]ico
10250	3	2023-01-03 00:00:00	1780	3	Antonio Moreno Taquería	Antonio Moreno	
10251	1	2023-01-04 00:00:00	600	1	Alfreds Futterkiste	Maria Anders	
10252	4	2023-01-05 00:00:00	380	4	Around the Horn	Thomas Hardy	[illegible]

Result

RIGHT JOIN Syntax

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

Note: The RIGHT JOIN keyword returns all records from the right table (customers),even if there are no matches in the left table

```
49 • SELECT * FROM  cs_order
50   RIGHT JOIN customers
51   ON cs_order.CustomerID = customers.CustomerID;
52
```

QUARY

Result

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

OrderID	CustomerID	OrderDate	Amount	CustomerID	CustomerName	ContactName	Country
10269	1	2023-01-22 00:00:00	1300	1	Alfreds Futterkiste	Maria Anders	Germany
10251	1	2023-01-04 00:00:00	600	1	Alfreds Futterkiste	Maria Anders	Germany
10248	1	2023-01-01 00:00:00	440	1	Alfreds Futterkiste	Maria Anders	Germany
10270	2	2023-01-23 00:00:00	1500	2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
10249	2	2023-01-02 00:00:00	1866	2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico

Result 4

INNER JOIN Syntax

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

The INNER JOIN keyword selects records that have matching values in both tables.

```
59 • SELECT *
60   FROM cs_order
61   INNER JOIN customers
62   ON cs_order.CustomerID= customers.CustomerID;
```

QUARY

Result

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

OrderID	CustomerID	OrderDate	Amount	CustomerID	CustomerName	ContactName	Country
10248	1	2023-01-01 00:00:00	440	1	Alfreds Futterkiste	Maria Anders	Germany
10249	2	2023-01-02 00:00:00	1866	2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
10250	3	2023-01-03 00:00:00	1780	3	Antonio Moreno Taquería	Antonio Moreno	Mexico
10251	1	2023-01-04 00:00:00	600	1	Alfreds Futterkiste	Maria Anders	Germany
10252	4	2023-01-05 00:00:00	380	4	Around the Horn	Thomas Hardy	UK
10253	5	2023-01-06 00:00:00	1200	5	Berglunds snabbköp	Christina Berglund	Sweden
10254	6	2023-01-07 00:00:00	800	6	Blauer See Delikatessen	Hanna Moos	Germany
10255	7	2023-01-08 00:00:00	500	7	Blondel père et fils	Frédérique Citeaux	France
10256	8	2023-01-09 00:00:00	1400	8	Bólido Comidas preparadas	Martín Sommer	Spain
10257	9	2023-01-10 00:00:00	900	9	Bon app'	Laurence Lebihan	France
10258	10	2023-01-11 00:00:00	2000	10	Bottom-Dollar Markets	Elizabeth Lincoln	Canada

Full join

```
SELECT *FROM table1
LEFT JOIN table2
ON
 table1.column_name = table2.column_name
UNION
SELECT *FROM table1
RIGHT JOIN table2
 ON
table1.column_name = table2.column_name;
```

Note: MySQL does not explicitly support a FULL OUTER JOIN. Instead, we can achieve it by combining a LEFT JOIN, a RIGHT JOIN, and a UNION operator. We can use FULL OUTER JOIN in SQL using the FULL OUTER JOIN keyword.

QUARY

```
SELECT * FROM
    cs_order
        LEFT JOIN  customers
        ON cs_order.CustomerID = customers.CustomerID
UNION
SELECT * FROM
    cs_order
        RIGHT JOIN customers
        ON cs_order.CustomerID = customers.CustomerID;
```

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

OrderID	CustomerID	OrderDate	Amount	CustomerID	CustomerName	ContactName
10248	1	2023-01-01 00:00:00	440	1	Alfreds Futterkiste	Maria Anders
10249	2	2023-01-02 00:00:00	1866	2	Ana Trujillo Emparedados y helados	Ana Trujillo
10250	3	2023-01-03 00:00:00	1780	3	Antonio Moreno Taquería	Antonio Moreno
10251	1	2023-01-04 00:00:00	600	1	Alfreds Futterkiste	Maria Anders
10252	4	2023-01-05 00:00:00	380	4	Around the Horn	Thomas Hardy
10253	5	2023-01-06 00:00:00	1200	5	Berglunds snabbköp	Christina Berglun
10254	6	2023-01-07 00:00:00	800	6	Blauer See Delikatessen	Hanna Moos
10255	7	2023-01-08 00:00:00	500	7	Blondel père et fils	Frédérique Citea
10256	8	2023-01-09 00:00:00	1400	8	Bólido Comidas preparadas	Martín Sommer
10257	9	2023-01-10 00:00:00	900	9	Bon app'	Laurence Lebihar
10258	10	2023-01-11 00:00:00	2000	10	Bottom-Dollar Markets	Elizabeth Lincoln
10259	11	2023-01-12 00:00:00	1000	11	Cactus Comidas para llevar	Patricio Simpson
10260	12	2023-01-13 00:00:00	1500	12	Centro comercial Moctezuma	Francisco Chang
10261	13	2023-01-14 00:00:00	750	13	Chop-suey Chinese	Yang Wang
10262	14	2023-01-15 00:00:00	1350	14	Comércio Mineiro	Pedro Afonso
10263	15	2023-01-16 00:00:00	1100	15	Consolidated Holdings	Elizabeth Brown
10264	16	2023-01-17 00:00:00	950	16	Dahlia Flowers Shop	Timothy Carter
10265	17	2023-01-18 00:00:00	500	17	Eastern Connection	Ann Devon

RESULT

6.Calculate the total order amount for each customer.

```
-- 6)   Calculate the total ord
SELECT CustomerID, SUM(Amount)
FROM cs_order
GROUP BY CustomerID
ORDER BY OrderAmount DESC;
```

QUARY

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

CustomerID	OrderAmount
3	3530
2	3366
5	2850
8	2600
9	2500
1	2340
6	2150
7	2050
10	2000
4	1830
12	1500
14	1350
18	1250

RESULT

7.Apply appropriate filter to show customers who have placed orders with an amount greater than $1000.

```
-- 7)   Apply appropriate filter to show customers
-- who have placed orders with an amount greater than $
SELECT CustomerID, SUM(Amount) as 'Order Amount'
FROM cs_order
GROUP BY CustomerID
HAVING SUM(Amount) > 1000;
```

QUARY

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

CustomerID	Order Amount
1	2340
2	3366
3	3530
4	1830
5	2850
6	2150
7	2050
8	2600
9	2500
10	2000
12	1500
14	1350
15	1100

RESULT

8.List all orders by each customer in descending order of order amount.

```
-- 8)   List all orders by each customer in descendi
select DISTINCT CustomerID , SUM(Amount)
FROM cs_order
GROUP BY CustomerID;
```

QUARY

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

CustomerID	SUM(Amount)
1	2340
2	3366
3	3530
4	1830
5	2850
6	2150
7	2050
8	2600
9	2500
10	2000

RESULT

9.Join with date filtering: Select orders placed in January 2023. (This is again to challenge you. It requires you to apply a combination of things we covered in class: Joins and filters)

```
-- 9)   Join with date filtering: Select orders placed in January 2023.
SELECT o.OrderDate,o.Amount,c.CustomerName
FROM cs_order o
JOIN
customers c
WHERE OrderDate  BETWEEN '2023-01-01' AND '2023-01-31';
```

QUARY

Result Grid | Filter Rows: | Export:

OrderDate	Amount	CustomerName
2023-01-30 00:00:00	1600	Alfreds Futterkiste
2023-01-29 00:00:00	1200	Alfreds Futterkiste
2023-01-28 00:00:00	1550	Alfreds Futterkiste
2023-01-27 00:00:00	1350	Alfreds Futterkiste
2023-01-26 00:00:00	1650	Alfreds Futterkiste
2023-01-25 00:00:00	1450	Alfreds Futterkiste
2023-01-24 00:00:00	1750	Alfreds Futterkiste
2023-01-23 00:00:00	1500	Alfreds Futterkiste
2023-01-22 00:00:00	1300	Alfreds Futterkiste
2023-01-21 00:00:00	850	Alfreds Futterkiste
2023-01-20 00:00:00	1050	Alfreds Futterkiste
2023-01-19 00:00:00	1250	Alfreds Futterkiste
2023-01-18 00:00:00	500	Alfreds Futterkiste
2023-01-17 00:00:00	950	Alfreds Futterkiste
2023-01-16 00:00:00	1100	Alfreds Futterkiste
2023-01-15 00:00:00	1350	Alfreds Futterkiste
2023-01-14 00:00:00	750	Alfreds Futterkiste
2023-01-13 00:00:00	1500	Alfreds Futterkiste
2023-01-12 00:00:00	1000	Alfreds Futterkiste

10.Write a brief report to go in your portfolio explaining your work and how you worked in a team.

Report: Database and SQL Task

Overview

In this project, our team was tasked with converting Excel files to CSV format, creating a new database, and importing the data into SQL Workbench. We then performed a series of SQL operations, including various types of joins, calculations, and filtering. The final deliverables included SQL queries, screenshots of the outputs, and a brief report to be included in our portfolios.

Team Collaboration and Task Breakdown

Our team initially divided the tasks to ensure efficient workflow. We decided to work together on the first few steps, which involved converting the Excel files into CSV format, creating the database, and uploading the data. This collaborative approach allowed us to troubleshoot issues as they arose and ensured that everyone was on the same page before moving on to more complex SQL queries.

Joint Efforts

1. File Conversion and Importing:

We worked together to convert the “Customers” and “Orders” Excel files into CSV format. Some team members faced challenges during this process, particularly with encoding errors when importing CSV files into SQL Workbench. To overcome this, we researched and found that converting CSV files to JSON could resolve the issue. We then successfully imported the JSON files into the database.

Challenge: A few members encountered difficulties importing the files. We assisted each other by sharing solutions and resources, such as links and tutorials, to guide those struggling with the task.

2. Database Creation and Data Upload:

The entire team participated in creating a new database in SQL Workbench and uploading the converted CSV (or JSON) files. We viewed the table structures and contents together, ensuring that the data was correctly imported and formatted.

3. SQL Joins:

We collaborated to apply various types of joins (Simple, Inner, Left, Right, Full) on the "Customers" and "Orders" tables. This was relatively straightforward for me and did not take much time, but some team members required extra assistance. We held group discussions to explain the concepts and syntax, which helped everyone understand and execute the joins successfully.

After completing the joint tasks related to the SQL joins, we each proceeded to work on the remaining tasks independently. This approach allowed each team member to focus on developing their SQL skills at their own pace while tackling the specific challenges of the project.

1. My Contributions:

I handled the tasks 6 to 9 independently. These required a deeper understanding of SQL functions and commands, which I achieved through a combination of prior knowledge, Google searches, and a trial-and-error approach.

2. Challenges and Solutions:

While the earlier tasks were relatively easy, the calculation and filtering steps were more challenging. I faced some difficulties in crafting the correct SQL queries, but by experimenting with different approaches and consulting online resources, I was able to complete the tasks successfully.

Conclusion

Despite some initial challenges, especially in understanding and importing the data, our team was able to successfully complete the project. The joint effort in the initial stages ensured that all team members were equipped with the necessary skills to move forward individually. The experience reinforced the importance of collaboration, resourcefulness, and persistence in solving complex problems. The final outputs, including the SQL queries and their results, were documented with appropriate screenshots and comments, which were included in our portfolios.

This project not only enhanced my SQL skills but also provided valuable experience in teamwork and problem-solving within a collaborative environment. Additionally, it deepened my understanding of database management and data manipulation, which are crucial skills for any data-

driven role. The challenges I faced and overcame have prepared me to tackle more complex database tasks in the future with greater confidence and efficiency.

BIBLIOGRAPHY

1. [Online] https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16.

2. [Online] https://cloud.google.com/learn/what-is-a-relational-database.

3. [Online] https://www.tutorialspoint.com/mysql/mysql_date_time_functions_timestampdiff.htm.

4. [Online] https://www.shiksha.com/online-courses/articles/primary-data-collection-methods-meaning-and-techniques/#:~:text=It%20is%20the%20data%20that,%2C%20observations%2C%20or%20direct%20measurements..

5. [Online] https://www.investopedia.com/terms/c/crowdsourcing.

6. [Online] https://www.qualitestgroup.com/assets/Crowdsourcing.pdf.

7. [Online] https://www.w3schools.com/mysql/mysql_join.asp.