SQL

data bases notes:

Structured Query Lang

Types of data Bases :

Relational

Form relations with one another

Using SQL

Non-Relational

The opposite

None-SQl

Will need a DMBS (Database Management System)

Using : MySQL

First:

CREATE DATABASE mydatabase; (to create a new data base)

USE mydatabase; (to make sure we are using the correct data base)

DROP DATABASE mydatabase; (to delete a database)

Altering data basses:

1. Setting a data base to (read only)

ALTER DATABASE mydatabase READ ONLY=1;

* When a database is on READ ONLY you can’t modify it but can access it , which means you cant DROP it .

TABLES:

Key words to create table called employes after we create the columns between the ();

After each column name+datatype we close with ,

1. Creating a table:



CREATE TABLE employes(

worker\_id INT,

INT for integers

Varchar( max character number)

Decimal( numbers before the comma , numbers after the comma)

first\_name varchar(50),

last\_name varchar(50),

hourly\_pay decimal(5,2),

hire\_date date

The last statement

We close with the type no more commas.

);

Closing the table

1. Selecting a table :

SELECT \*FROM employes;

1. Re-Naming a table :

RENAME TABLE employes TO NewName;

1. deleting a table:

DROP TABLE employes;

1. altering a table :

NO semi-colon after the first line!

-Adding a column

ALTER TABLE employes

ADD phone\_number VARCHAR(1G);

-Renaming a column

ALTER TABLE employes

RENAME COLUMN phone\_number TO email ;

-changing the type

ALTER TABLE employes

MODIFY COLUMN email VARCHAR(100);

-changing the placement of a column :

ALTER TABLE employes

MODIFY email VARCHAR(100)

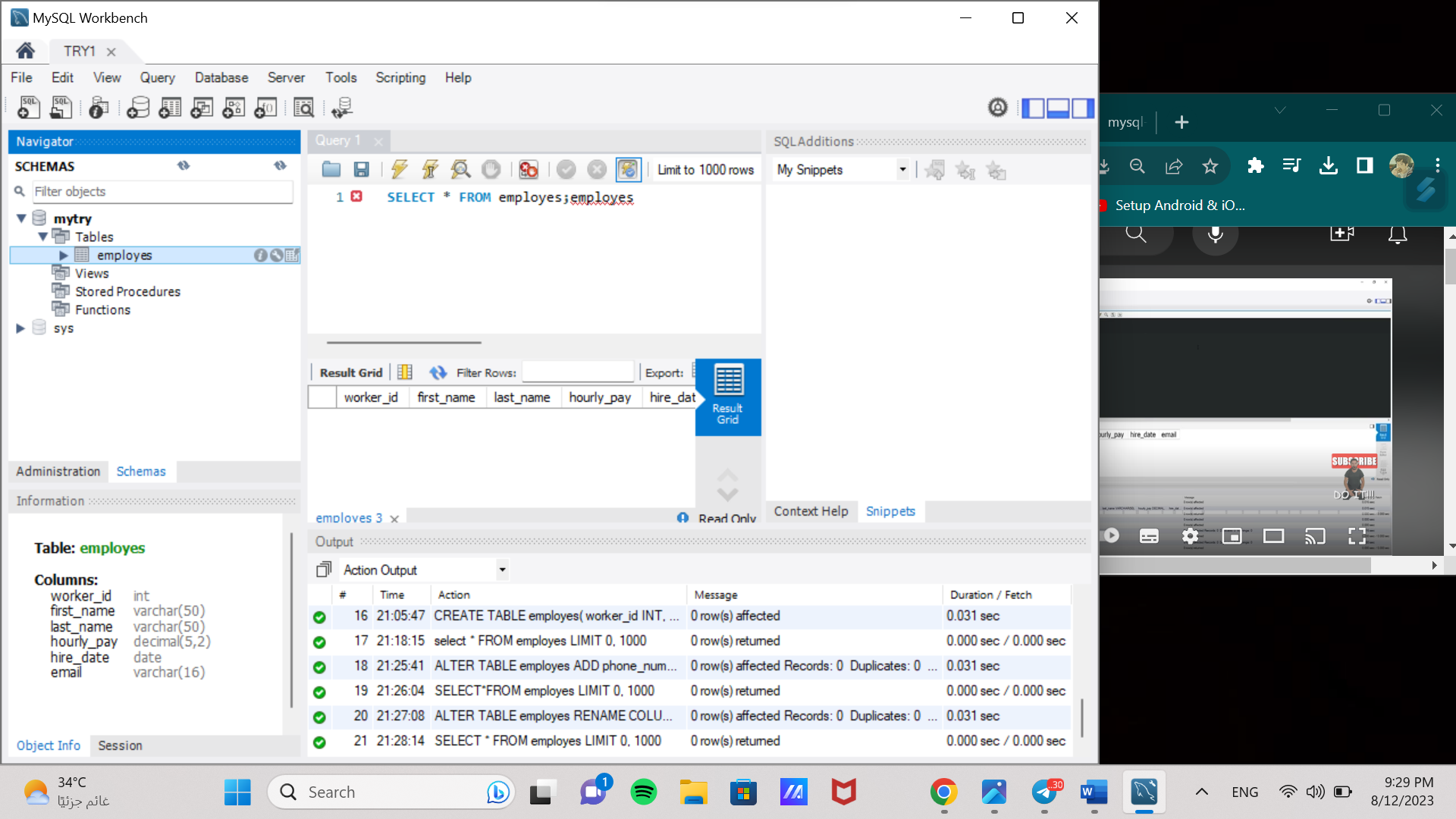
AFTER last\_name;

-to make it the first column

ALTER TABLE employes

MODIFY email VARCHAR(100)

FIRST;



# can check the data types for columns here

* Rows:

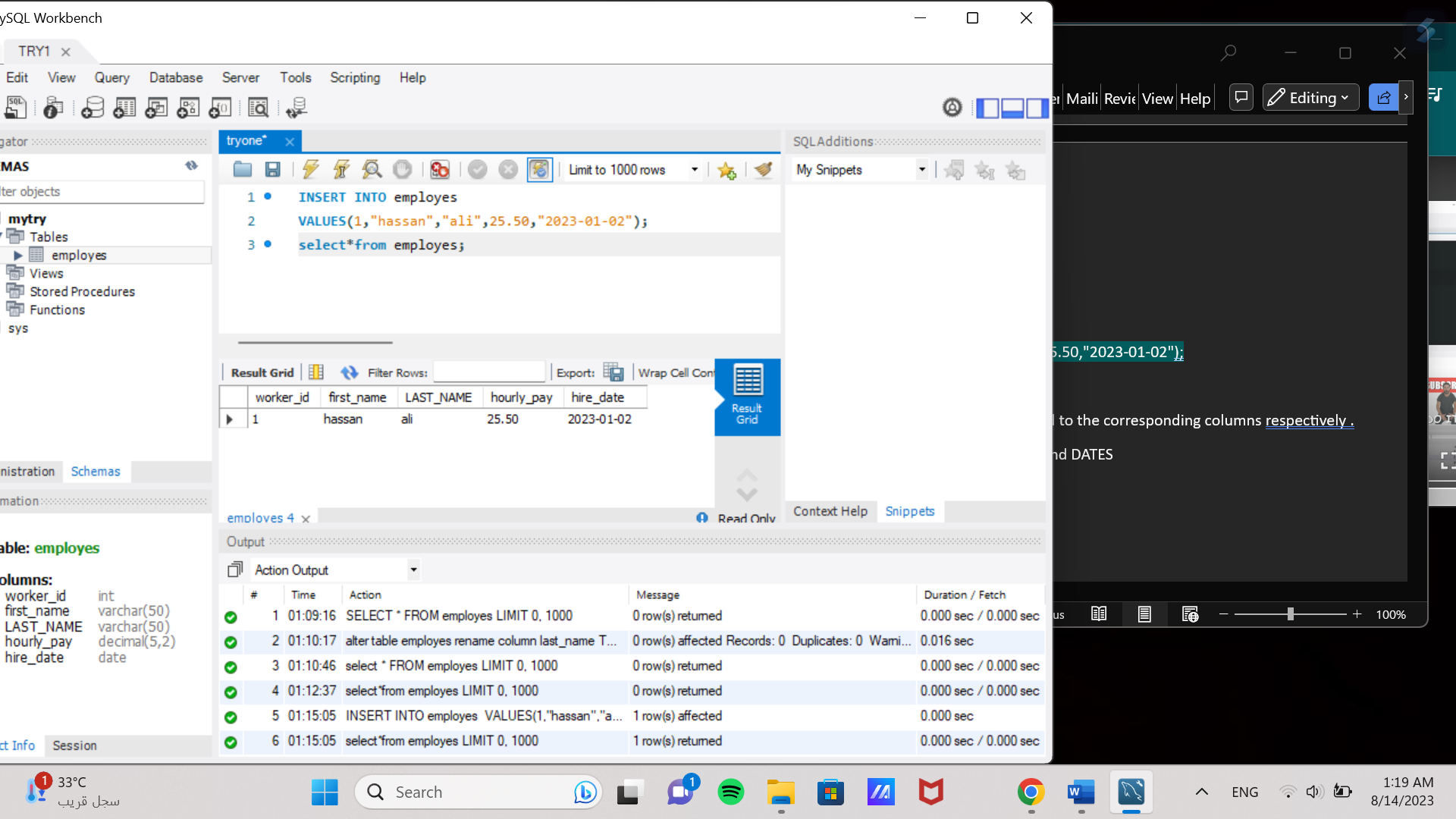
INSERT INTO employes

VALUES(1,"hassan","ali",25.50,"2023-01-02");

These values will be added to the corresponding columns respectively .

We use “ ” for VARCHAR and DATES

Example :



* Adding multiple lines of rows at once

INSERT INTO employes

VALUES(1,"hassan","ali",25.50,"2023-01-02"),

(2,"ahmad","hassan",4.3,"2023-06-01"),

(3,"bo","hassan",4.3,"2023-09-01"),

(4,"hashim","hassan",4.3,"2023-02-01");

NOTE : after those lines of codes are executed, executing them will add a repeated row

* How to delete a row :
* Delete ALL rows :

DELETE FROM employes;

-delete a specific row :

SET SQL\_SAFE\_UPDATES=0;

DELETE FROM employes

WHERE worker\_id=1;

SELECT\*FROM employes;

SET SQL\_SAFE\_UPDATES=1;

This will delete the rows where the worker\_id =1 ,the where clues is really important.

* How to add data to a row without filling all required columns

INSERT INTO employes (worker\_id,first\_name,last\_name)

VALUES(127,"mark","lee");

* Retrieving data

1. Showing the whole table :

SELECT\*FROM employes;

1. Retrieving specific columns

SELECT first\_name,LAST\_NAME FROM employes;

1. The WHERE clues

SELECT \*FROM employes

WHERE worker\_id=1;

This will display every row from the table employes where the worker\_id is the number 1;

Ex2;

SELECT \*FROM employes

WHERE first\_name="hassan";

Ex3;

SELECT\*FROM employes

WHERE hourly\_pay>=25;

Ex4 (checking if something is NULL);

SELECT\*FROM employes

WHERE hire\_date IS NULL ; ( we could also use NOT NULL for the opposite )

We can use this with dates as well . we can check if something is not equal by using !=

* How to update data :

SET SQL\_SAFE\_UPDATES=0; // to prevent an error 1175

UPDATE employes

SET hire\_date="2023-02-15"

WHERE worker\_id=127; //to pinpoint the exact location of the change

SELECT\*FROM employes;

SET SQL\_SAFE\_UPDATES=1; // turning it back on

NOTE ; if we delete the line with the WHERE clue then we will update the chosen column for all rows simultaneously

* AUTOCOMMIT:

When working on a project lost or deleted data is gone forever upon executing because of auto commit being ON

So to turn it off we can do this :

1. SET AUTOCOMMIT=OFF;

Here autocommit is turned off , which means our work need to be saved manually .

1. COMMIT;

Adding this line will save our previous work ! after writing this line u cant retrieve deleted data

1. ROLLBACK;

This line will dismiss all actions made after the last COMMIT command !

* Accessing current date and time :

INSERT INTO tablename ( When creating the table we declared three variables , their types respectively is DATE,TIME,DATETIME . and now we are just filling them)

VALUES( CURRENT-DATE() , CURRENT\_TIME() ,NOW() ); ( those are built-in functions for the previously mentioned variables respectively, this is how we filled them in )

SELECT\*FROM tablename;

* The kyeWord UNIQUE

When we add UNIQUE after a line of code

For example : varible\_name VARCHAR(20) UNIQUE;

This ensures that the variable names under the UNIQUE column cant be repeated

If you forget to add it while creating the variable , you can add it later on like this :

ALTER TABLE tableName

ADD CONSTRAINT

UNIQUE(varible\_name);

* The NOT NULL keyword
* Adding not null in place of unique so the column cant carry a null row
* To add the condition to an already created column
* ALTER TABLE tableName
* MODIFY clumnName TypeOfColumn NOT NULL ;
* CHECK keyword :
* CHECK(column-name >= 10.00)
* HOWEEVER , we can create a stable check that we can drop anytime ,ex:
* CONSTRAINT check­­\_name CHECK(column\_name>=10.00);

-to add a check to an already existing table

ALTER TABLE tableName

ADD CONSTRAINT check­­\_name CHECK(column\_name>=10.00);

* To drop (stop using a check ) write :

ALTER TABLE tablename

DROP CHECK checkname ;

Th e default keyword :

* CREATE TABLE tablename(

Price DECIMAL (4,2) DEFALT 0; )

* To alter an already existing table
* ALTER TABLE tablename

ALTER Price SET DEFALT 0;

Primary key

* Can only have one in each table
* Allow us to eliminate duplicates under a column
* This make editing the table easier since each row will need to have at least one unique column

To use it while creating the table :  
CREATE TABLE tablename(

varibleName INT PRIMARY KEY

);

To add it later :

ALTER TABLE tableName

ADD CONSTRAINT

PRIMARY KEY( ColumnName);

* AUTO-INCREMENT
* Can be applied to a column that is set as a key
* So in the previous example , we can set in like this

CREATE TABLE tablename(

varibleName INT PRIMARY KEY AUTO\_INCREMENT

);

* So while adding other rows beside the prim key , it will be incremented regardless , will start from 1 and go on ( unless we set it to a diff start )
* Setting the auto-increment to a specific start :

ALTER TABLE tableName

AUTO­\_INCREMENT = 100;

* Foreign key
* When a primary key from a table is used in a different one
* Think of it as the link between different tables
* Ex: use it as a link between customer information and their transactions
* So, both tables should

FOREIGN KEY (the variable name in table A) REFERNCES tableB (the variable name with the primary key in table B);

* To delete a foreign key :

ALTER TABLE tableA

DROP FOREIGN KEY foreignKeyName ;

Note :(to find the name of the foreign key u can find it in the files under the table)

* To give ur foreign key a name:

ALTER TABLE tableA

ADD CONSTRAINT TheNameYouWant

FOREIGN KEY (ColumnName From TableA) REFERNCES TableB(Column name with the primary key)

Triggers:

Insert trigger:

CREATE TRIGGER name ON tablename

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON: /\*\* SO it doesn’t show when rows are effected, it will run in the background

DECLARE MYVARB TYPE() // that will equal the value to be inserted

SELECT MYVARB = INSERTED.(the inserted varible value from the mother table).

FROM INSERTED

INSERT INTO tablename (COLUMN NAME )

VALUES(MYVARB)

END