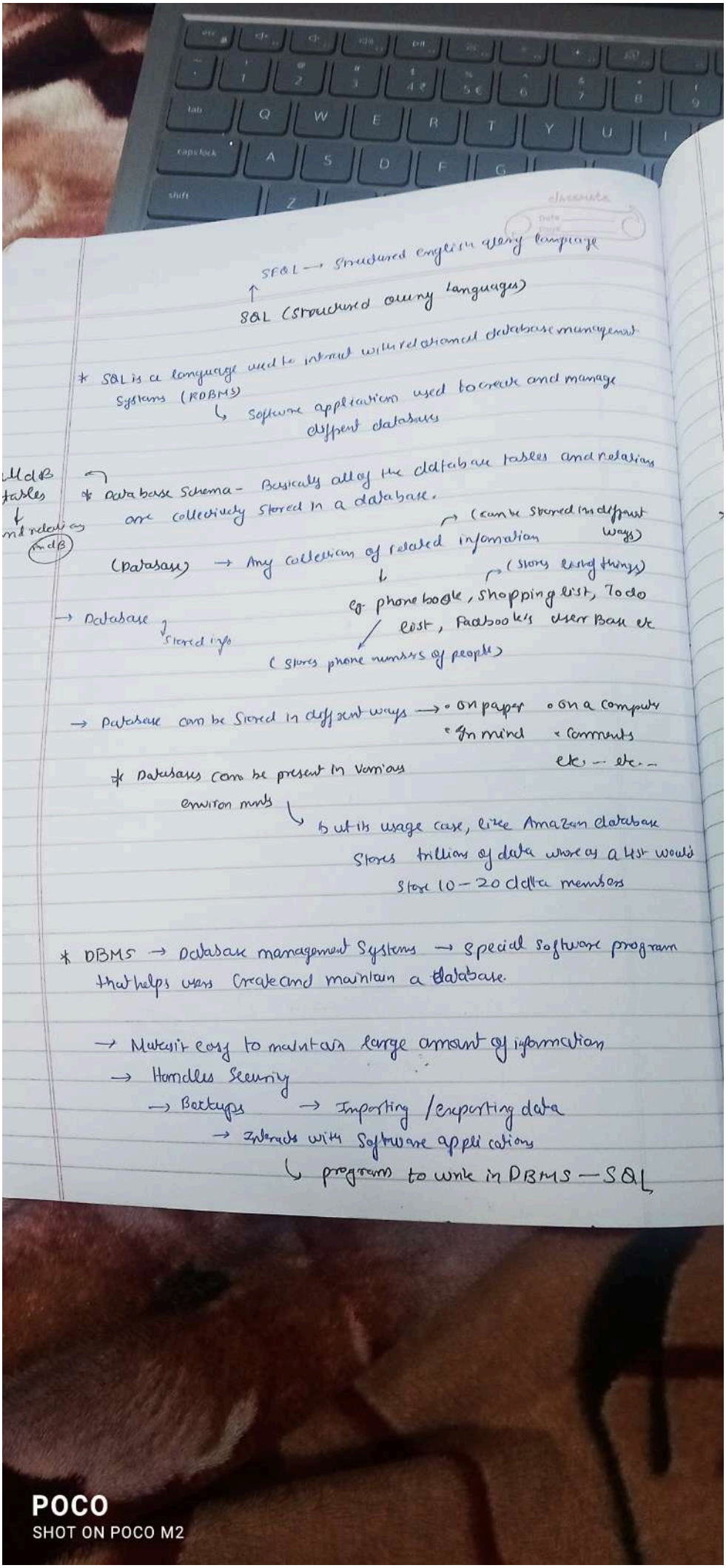
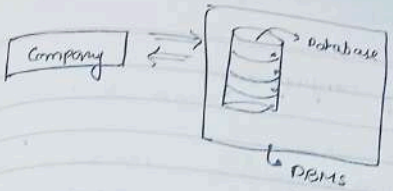




* Company for eg Amazon will interact with DBMS in order to create, delete and update information

CRUD → Create, read, update, delete

↳ we want our database to perform these 4 things in an effective manner to be called a good database

* Two types of Databases:

Relational Databases (SQL)	Non-Relational Databases (NoSQL)
• organizes data into one or more tables	• Organizes data is anything but a traditional table → not a relational database
* A table has columns & rows	→ Key-value stores → Graphs • Documents (JSON, XML etc) • Graphs, flexible tables
* A unique key identifies each row	↳ Storing of data in these databases
→ rows = tuples, columns = attributes	

Eg (Relational Database) → MySQL, POSTGRES etc

(STUDENT TABLE)

* ID #	Name	Major
1	Jack	Biology
2	Kate	Sociology
3	Claire	English
4	John	Chemistry

→ Here ID is acting as a key uniquely identifies each row in our table.

↳ It has unique value for every row

dummy → insertion
↳ check condition

stu-age interchecr (stu - age) = '17

* SQL (structured query language) → used to interact with RDBMS

performs CRUD operations as well as other tasks (user management, security, backups etc.)

→ used to define tables and structures

(Relational database management System)

↓

Helps us to create and maintain relational database

→ SQL code used on one RDBMS is not always portable to another without modification.

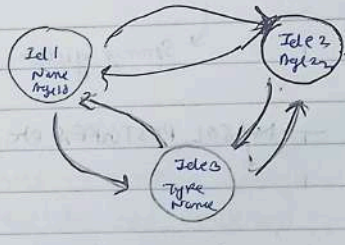
* Non-Relational Databases (NOSQL) → uses other data structures

↳ • storing of data in a non-relational manner in various forms

↓

• Document, Graphs, Key-Value Hash

↓ (by relational nodes)



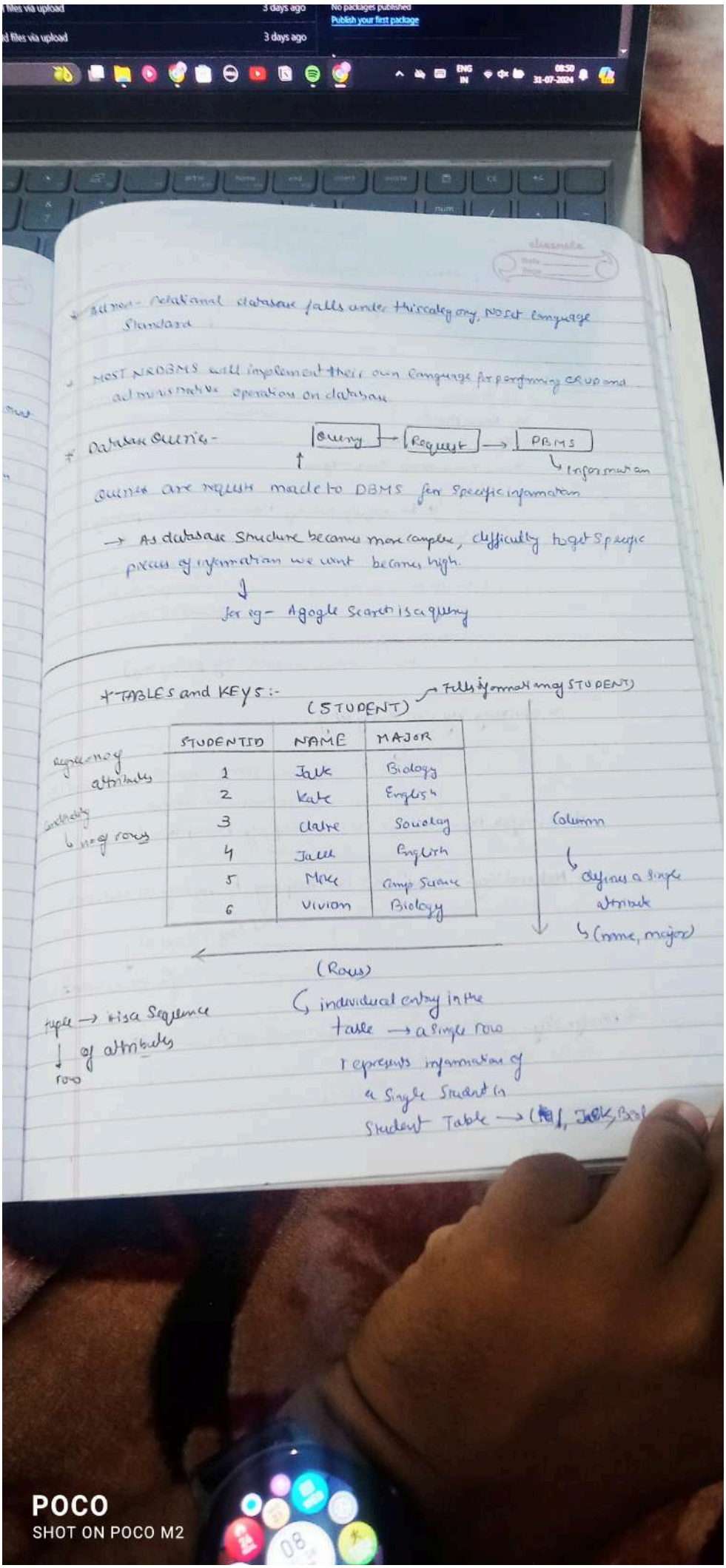
↓ (key values mapped)

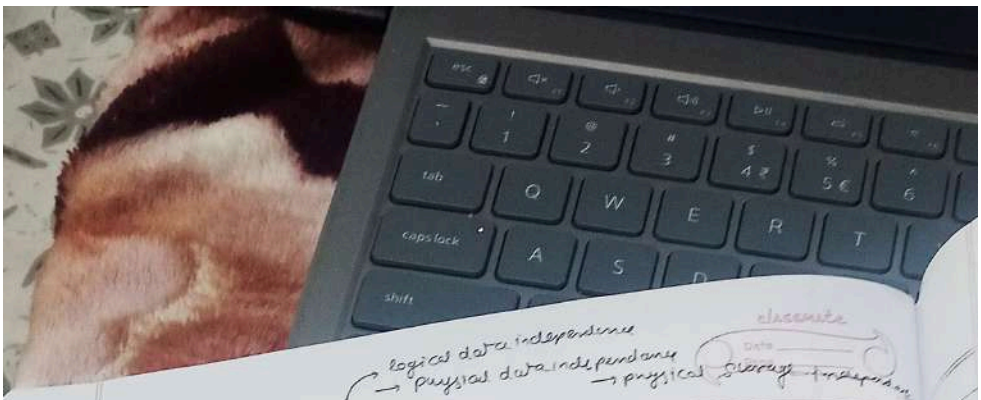
Key	Value
"xyz"	String
"abc"	JSON
"pqr"	BLOB
"1mno"	etc...

* Like RDBMS, we can have non-relational database management System (NRDBMS)

↳ Create and maintain non-relational database

↳ MONGODB, firebase etc.





→ logical data independence
→ physical data independence → physical storage independence
↑ views for data independence

* In RDBMS, we always need to have a special column
↓
primary key
↓
- we use it when ← (an attribute that uniquely defines row in a database)
we know that in a table we have entries with same name and same majors

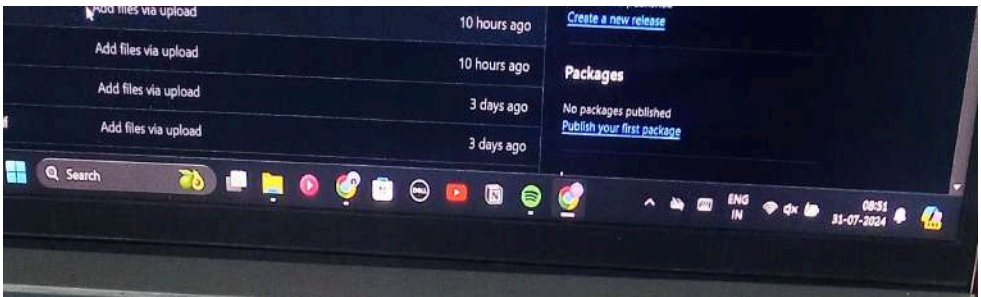
↳ to uniquely identify them → we have primary key
(to differentiate both of them)
↑
(we use primary key) ← (as value of primary key in every row is unique, therefore to identify elements in that row)
↓
(we can differentiate b/w 2 values in a table by primary key)

* primary key can be anything
↳ ◦ number, string, user id, user email etc
but it must be unique

* Surrogate Key - key that has no mapping to anything in real world

* Natural Key - A key that has mapping to anything in real world
↳ (eg. SSN id)
↓
for the whole world

* Foreign Key - Attribute which we can store on database table to store another database table
↳ stores



FK (foreign key)

a primary key in another table

empId	Name	branchId
100	Jan	1
101	Om	2
102	Josh	3

branchId	b name	mgrId
2	Scrap	101
3	Skum	102
1	Corpo	108

These can further relate to other tables by using foreign key

(FK → defines a relationship between two tables)

↳ which branch does a name belong to by using foreign key in employee table

* A particular table can have more than 1 branch elements

* Composite key = Any key with more than 1 attribute

branch id	Supplier name
2	Hammer Mill
2	Uniball
2	Patriot paper
3	Uniball
3	Hammer Mill

They both together can uniquely identify each row

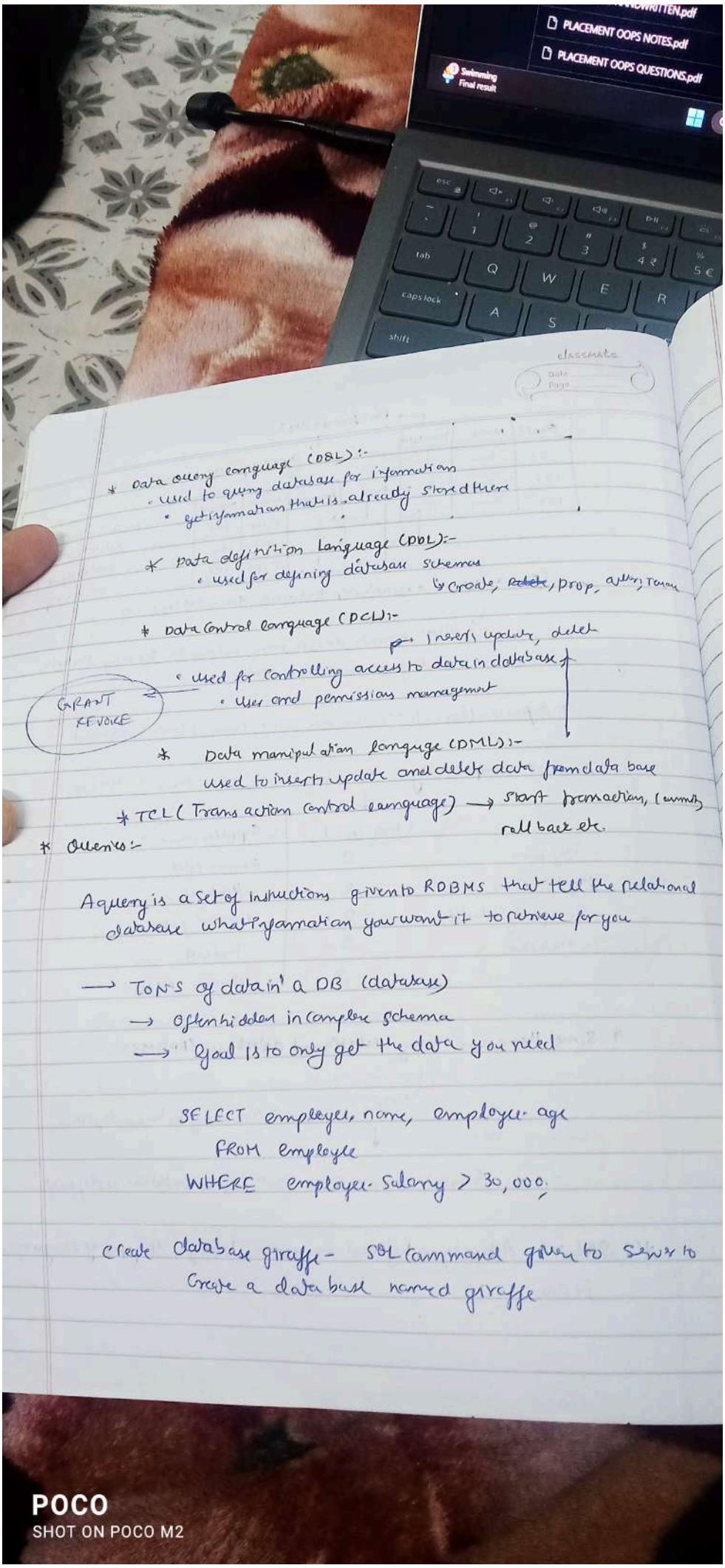
* Since there are many relational databases (RDBMS)

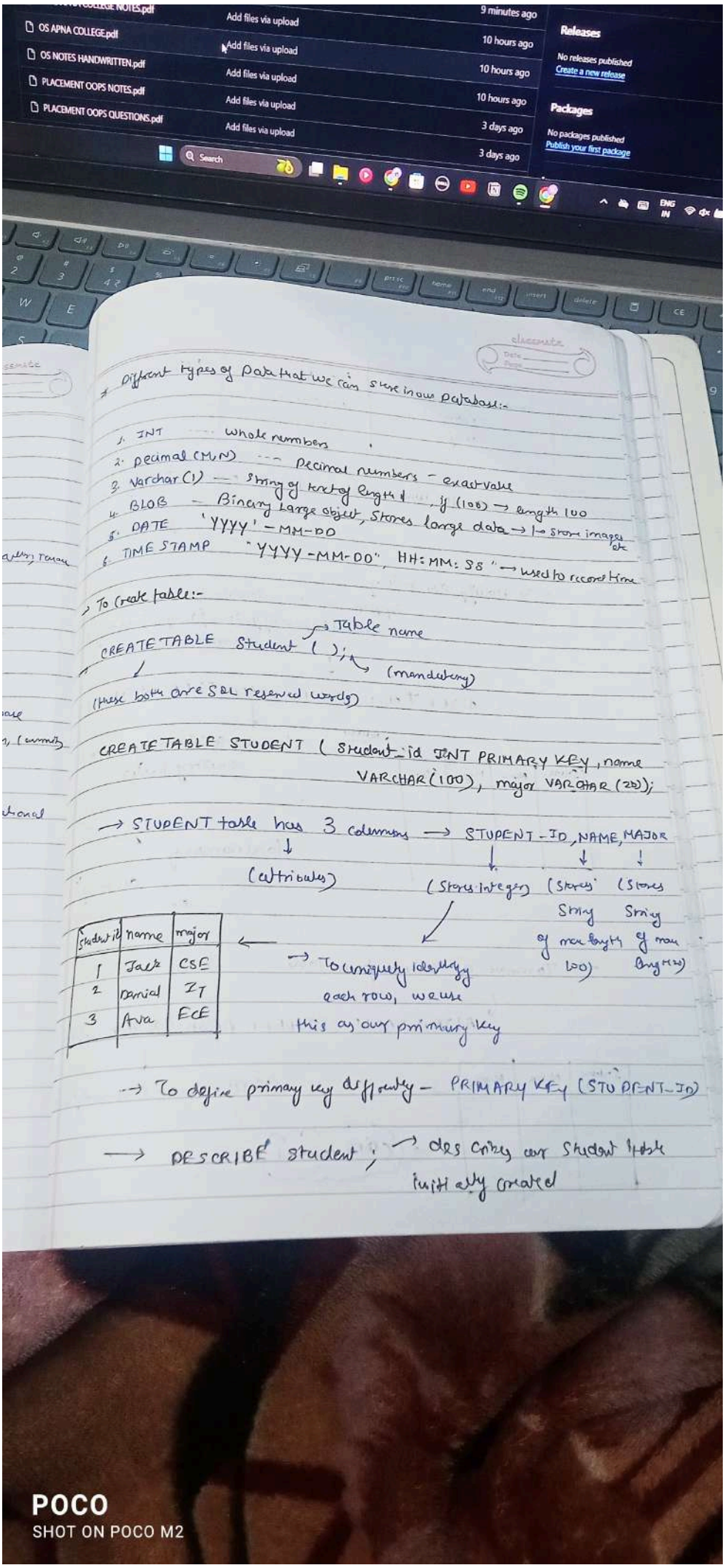
↳ SQL implementations vary between systems

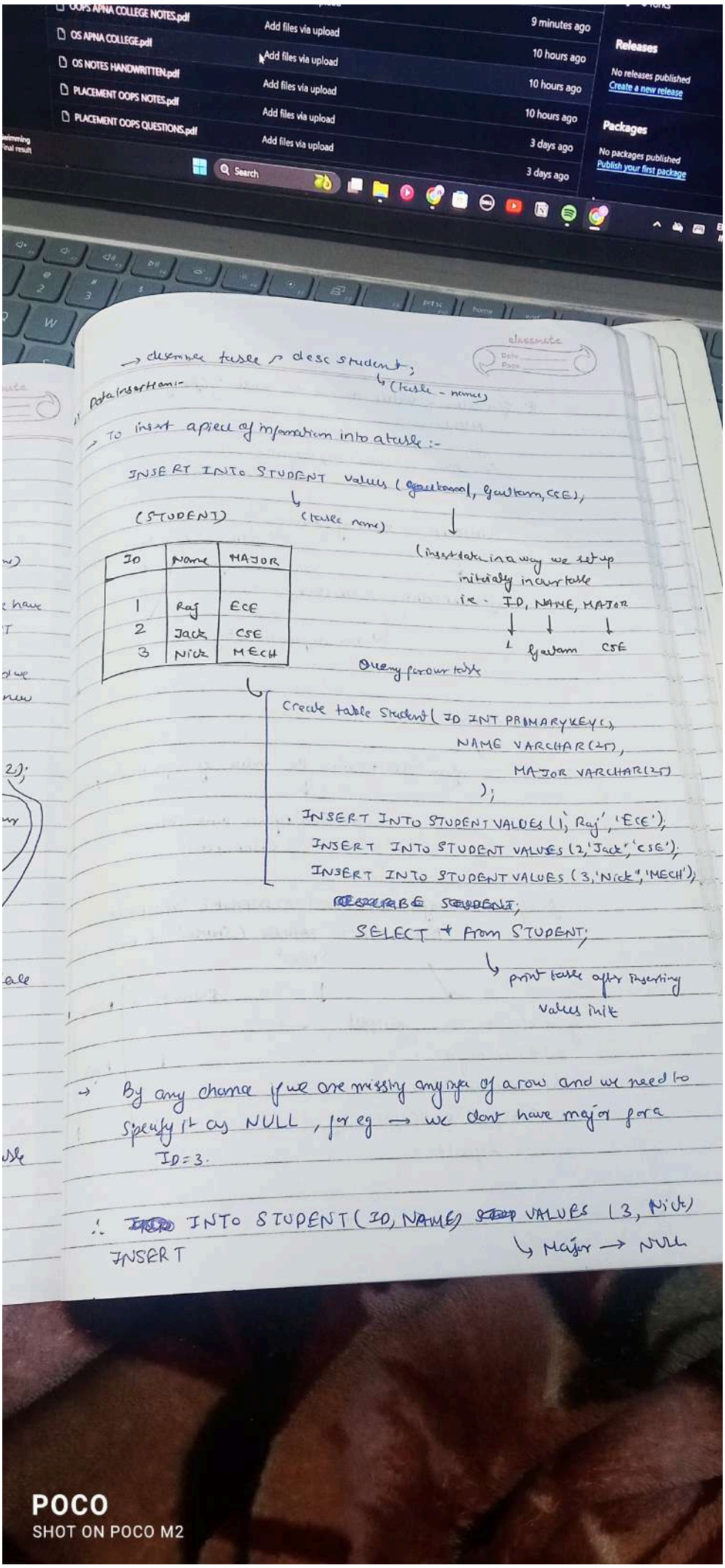
↓

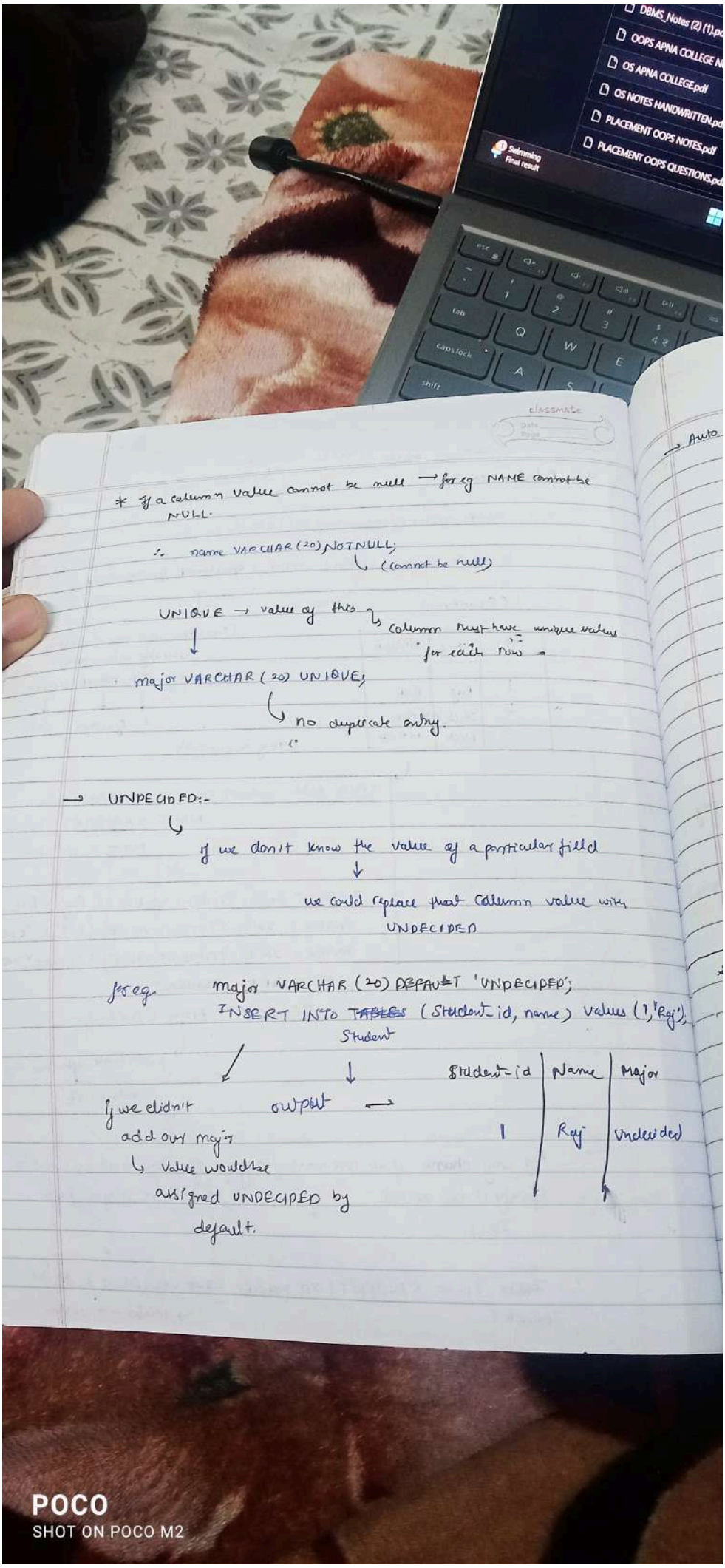
Concept remains same but implementation is different

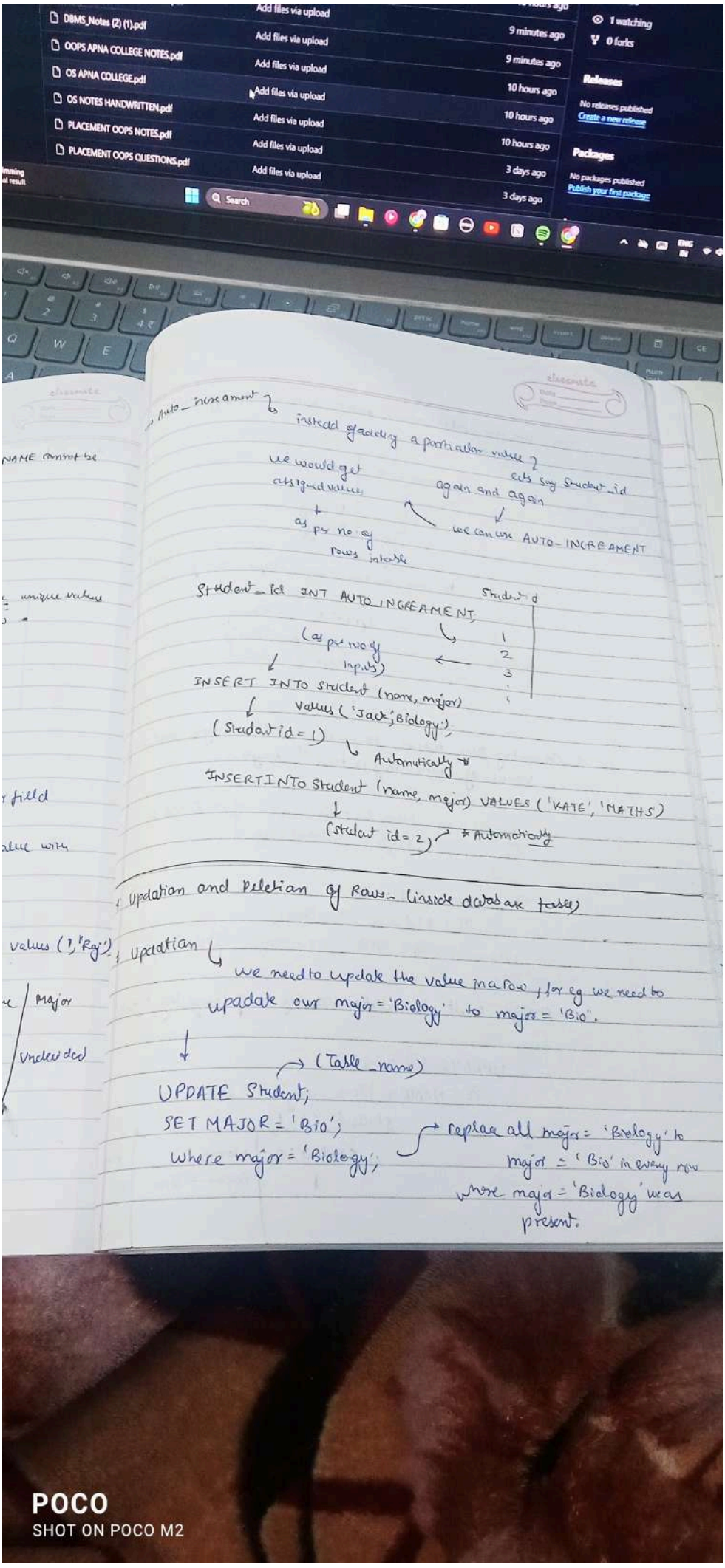
* SQL → Actually a hybrid language, having 4 types of languages in one.

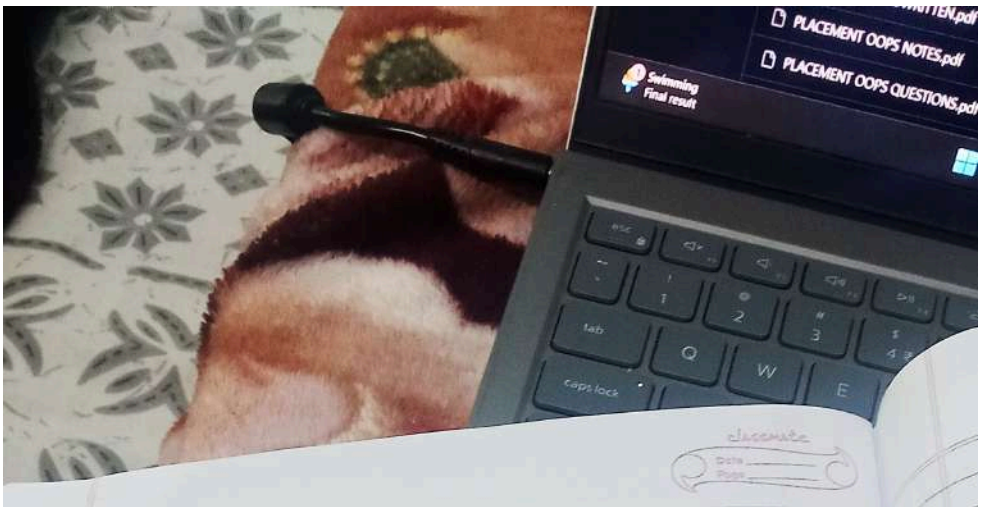












→ we can update a row's values without changing the values of other row with same value

i.e	student_id	major
	1	Biology
	4	Biology

↓

but only need to change Major = 'BIO' for Student_id = 4

↓

∴ UPDATE STUDENT
SET MAJOR = 'BIO';
where student_id = 4;

→ Combining two majors to result a 3rd major and changing values of both majors to 3rd major.

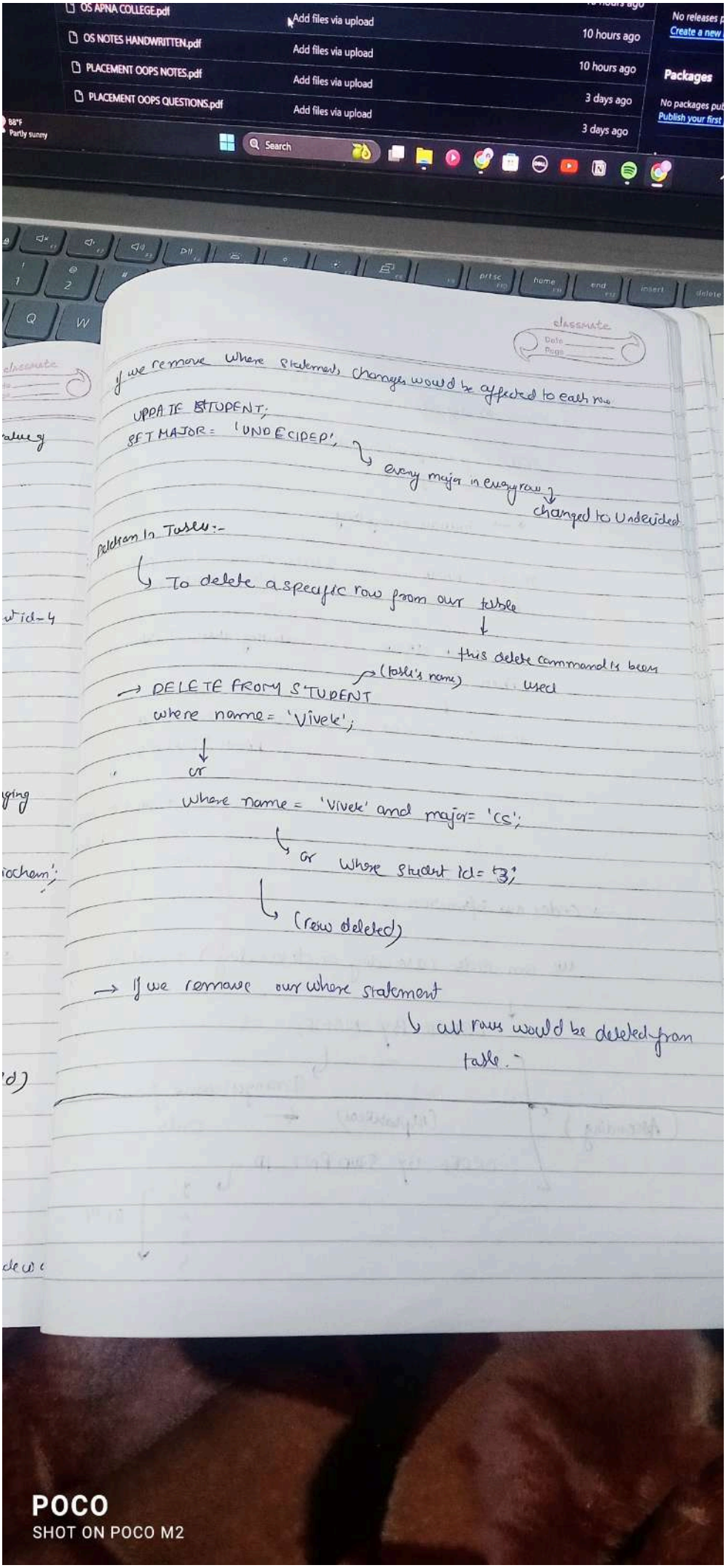
for eg Major 1 = 'BIO'; Major 2 = 'Chemistry' → Major 3 = 'Biochem';

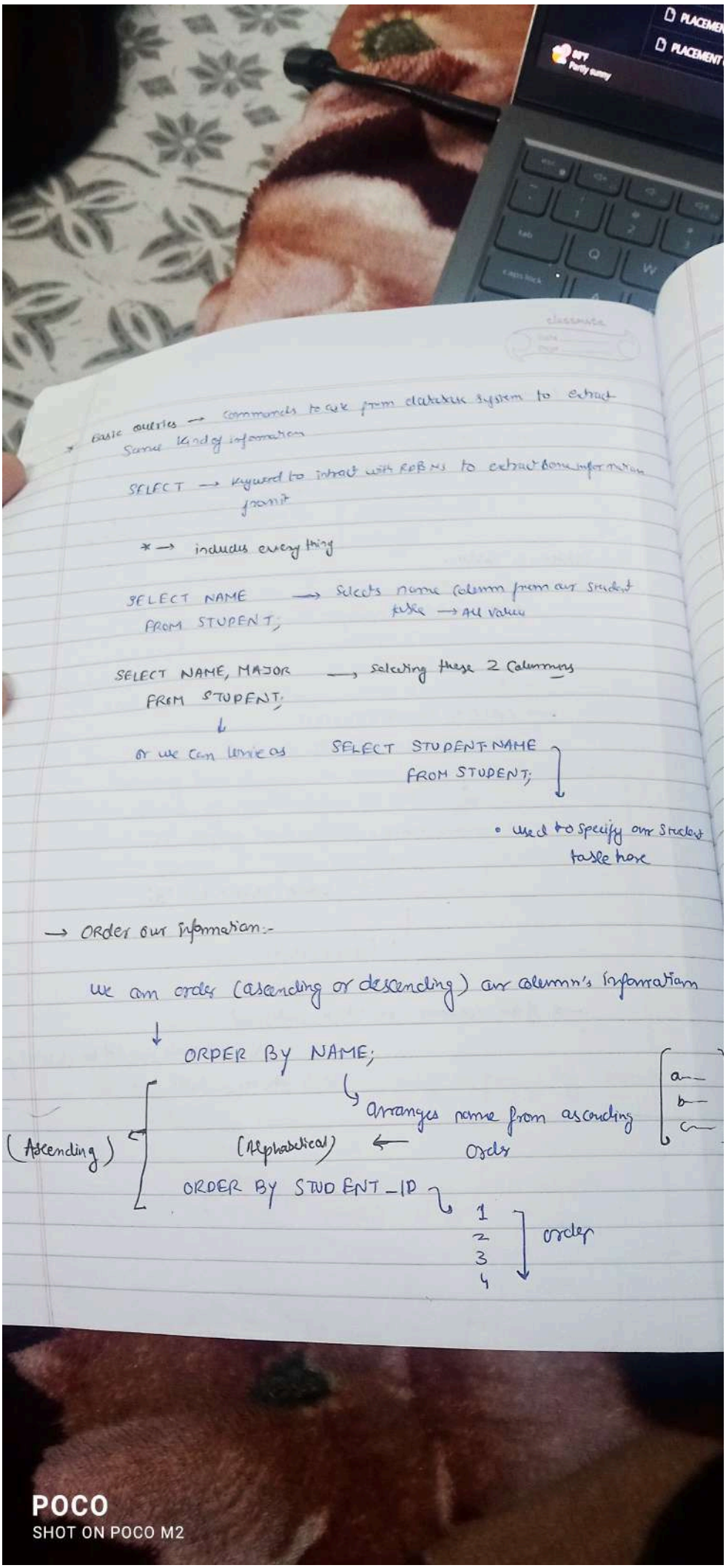
∴ UPDATE STUDENT
SET MAJOR = 'Biochem';
where major = 'BIO' or major = 'Chemistry';

→ change multiple values according to primary key (Student_id)

UPDATE STUDENT
SET NAME = 'Tom', major = 'UN DECIDED';
where student_id = 1;

Student id	Name	Major
1	Raj → Tom	BIO → Und





* Basic queries → commands to ask from database system to extract some kind of information

SELECT → keyword to interact with RDBMS to extract some information from it

* → includes everything

SELECT NAME → selects name column from our student table → All values
FROM STUDENT;

SELECT NAME, MAJOR → selecting these 2 columns
FROM STUDENT;

or we can write as SELECT STUDENT.NAME
FROM STUDENT;

• used to specify our student table here

→ ORDER our information:-

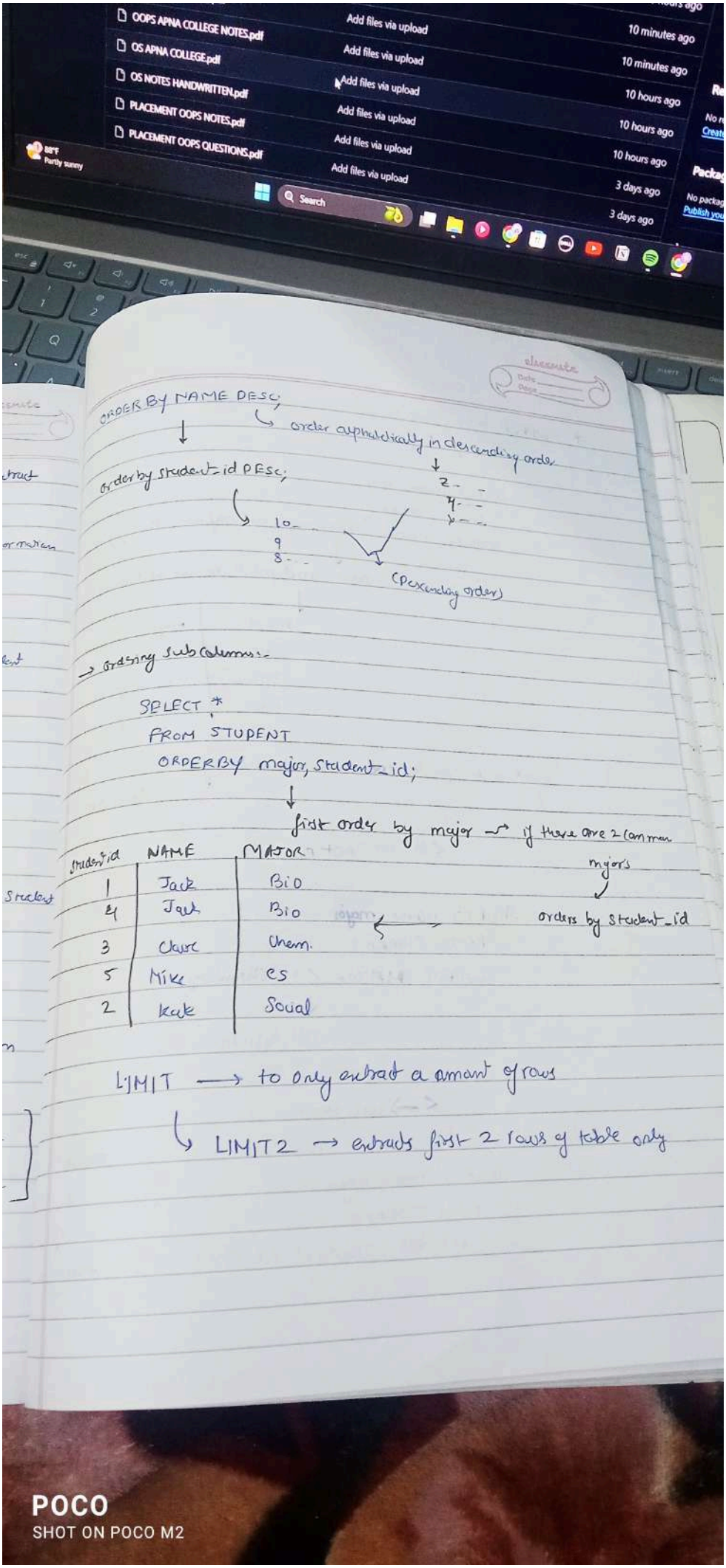
We can order (ascending or descending) our column's information

ORDER BY NAME;

Arranges name from ascending order (Alphabetical) ← (Ascending)

ORDER BY STUDENT_ID

1
2
3
4
order



* WHERE → required to select an information from table

```
SELECT name, major
  FROM STUDENT
WHERE major = 'chemistry' or major = "Biology";
```

↓

select and print names and major according

name	major
Jack	Biology
Clark	Chemistry
Jack	Biology

→ other operators we can make use of <, >, <=, >=, =, <>, AND, OR

<> → Not equal to.

```
SELECT name, major
  FROM STUDENT
WHERE MAJOR <> "Chemistry";
```

↓

not equal to

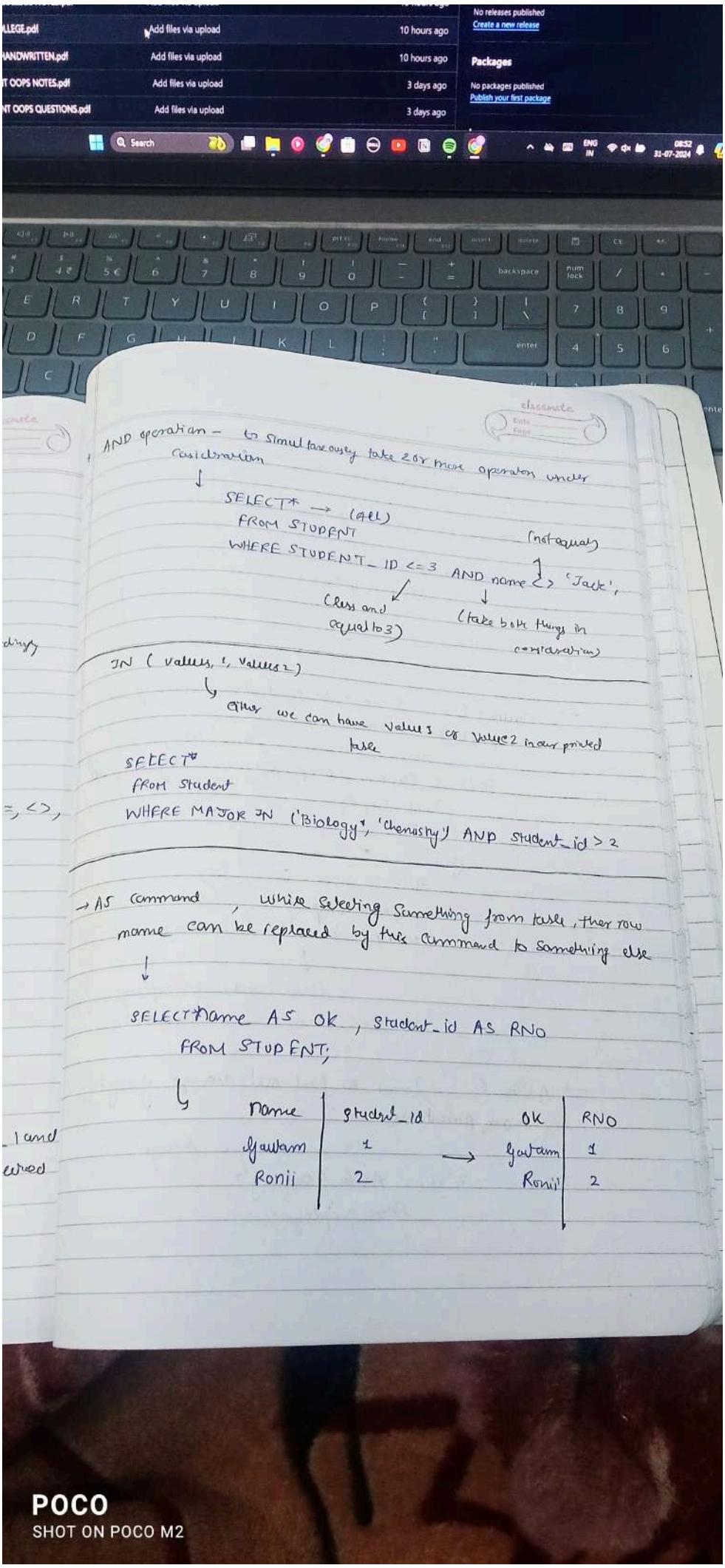
or <= → less than

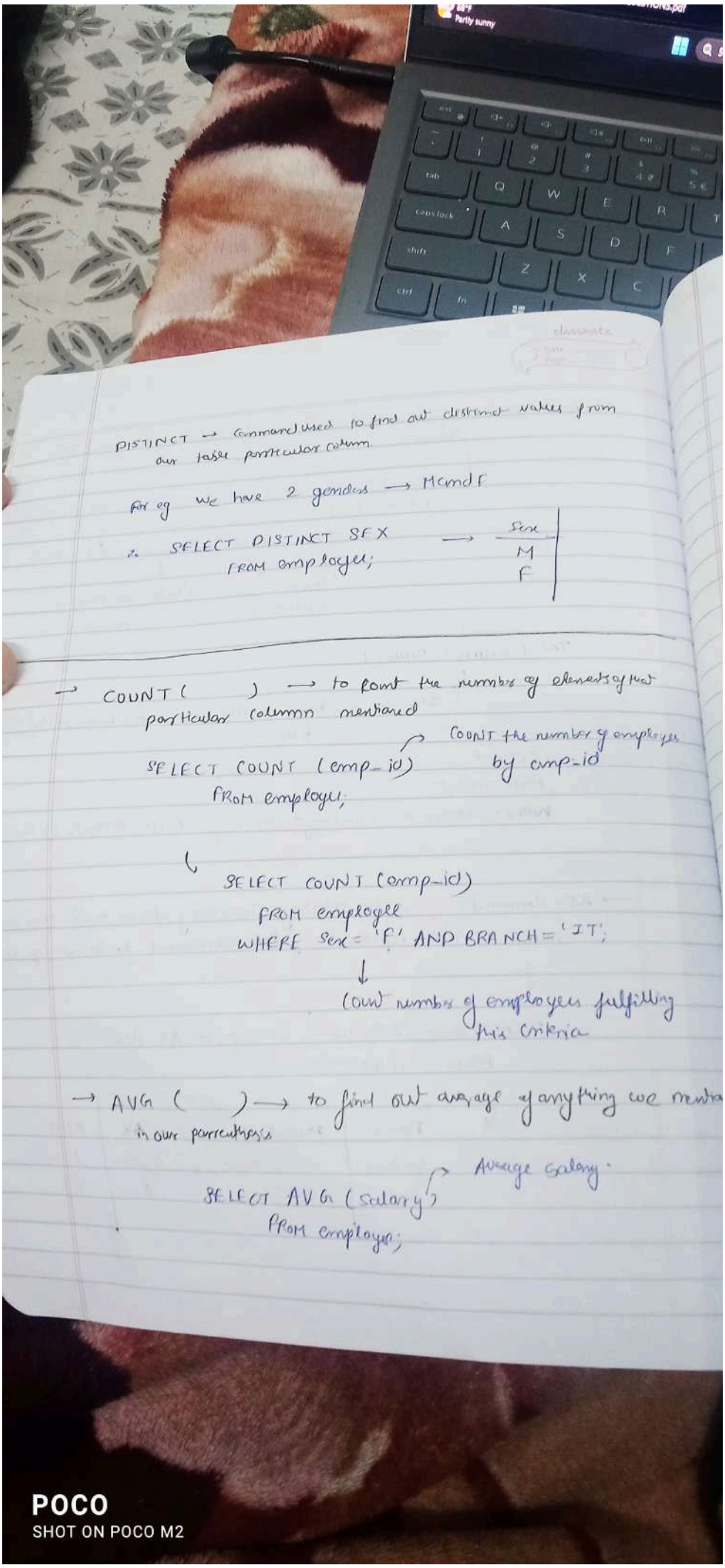
```
SELECT name, major
  FROM STUDENT
WHERE Student_Id <3;
```

student_id_1 and 2 selected

↓

these rows





DISTINCT → command used to find out distinct values from our table particular column.

for eg we have 2 genders → M and F

∴ SELECT DISTINCT SEX
FROM employee;

Sex
M
F

→ COUNT ( ) → to count the number of elements of that particular column mentioned

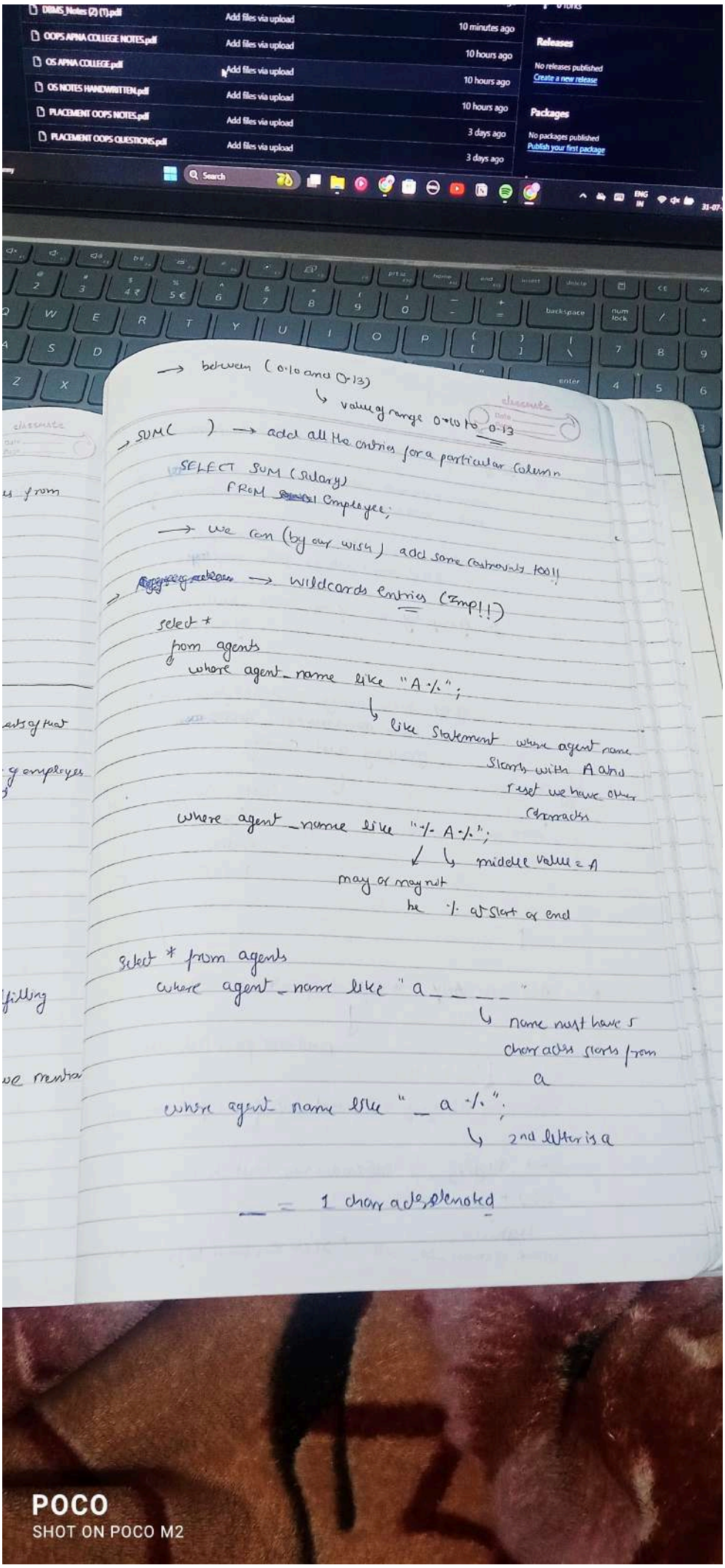
SELECT COUNT (emp_id) → COUNT the number of employees by emp-id
FROM employee;

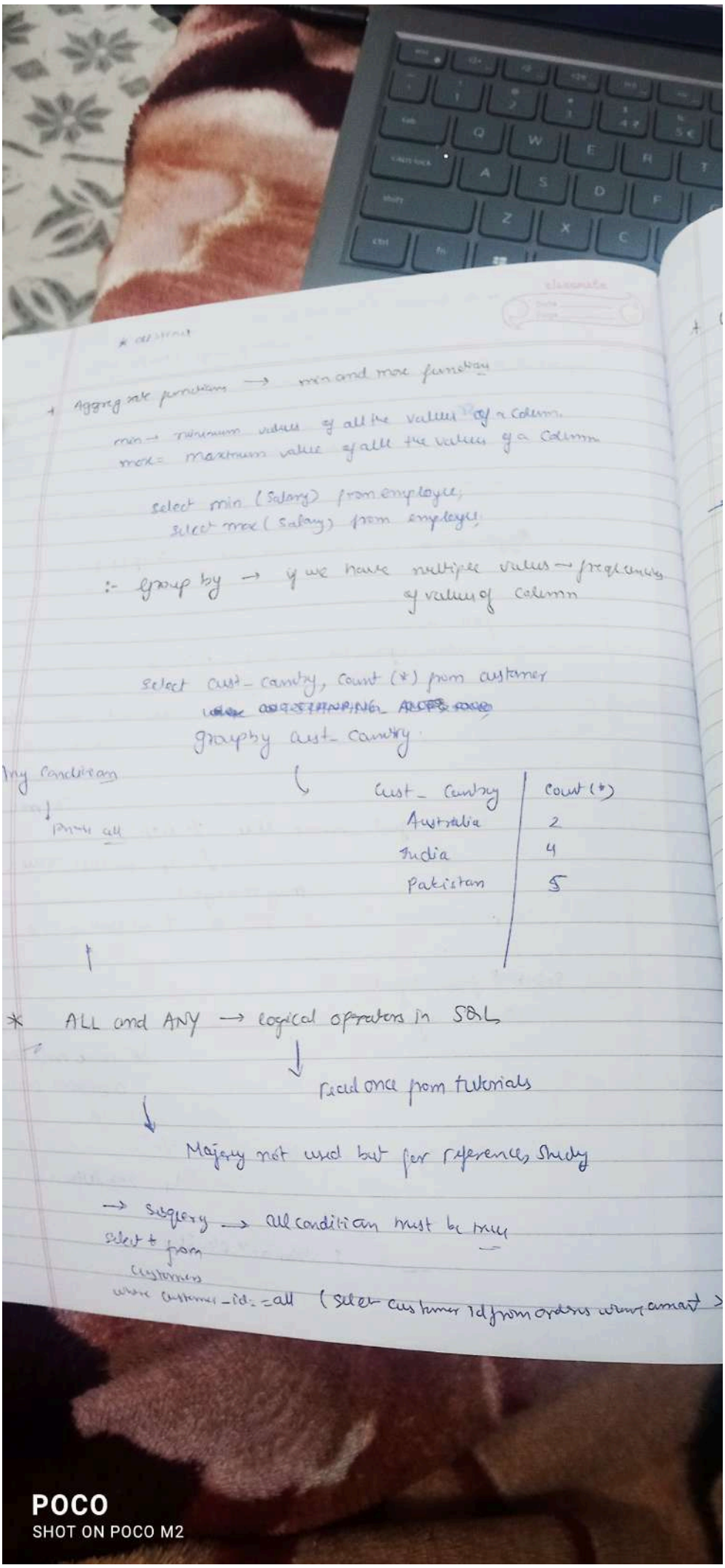
↳ SELECT COUNT (emp_id)
FROM employee
WHERE Sex = 'F' AND BRANCH = 'IT';
↓
(our) number of employees fulfilling this criteria

→ AVG ( ) → to find out average of anything we mention in over parenthesis

SELECT AVG (salary) → Average salary.
FROM employee;

POCO
SHOT ON POCO M2





+ Aggregate functions → min and max functions

min → minimum value of all the values of a column.
max = maximum value of all the values of a column

select min (Salary) from employee;
select max (Salary) from employee;

:- group by → if we have multiple values → frequency of values of column

select cust_country, count (*) from customer
group by cust_country;

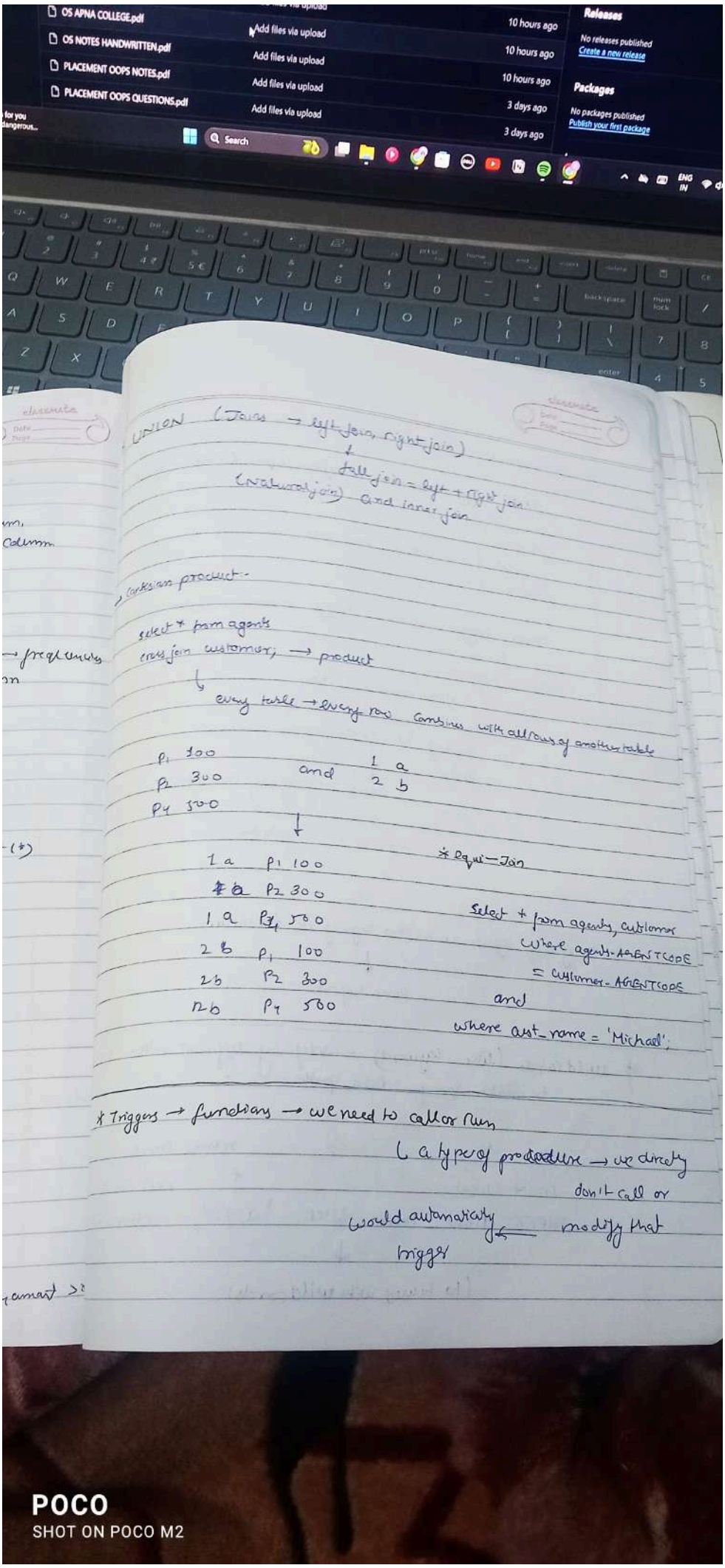
cust_country	count (*)
Australia	2
India	4
Pakistan	5

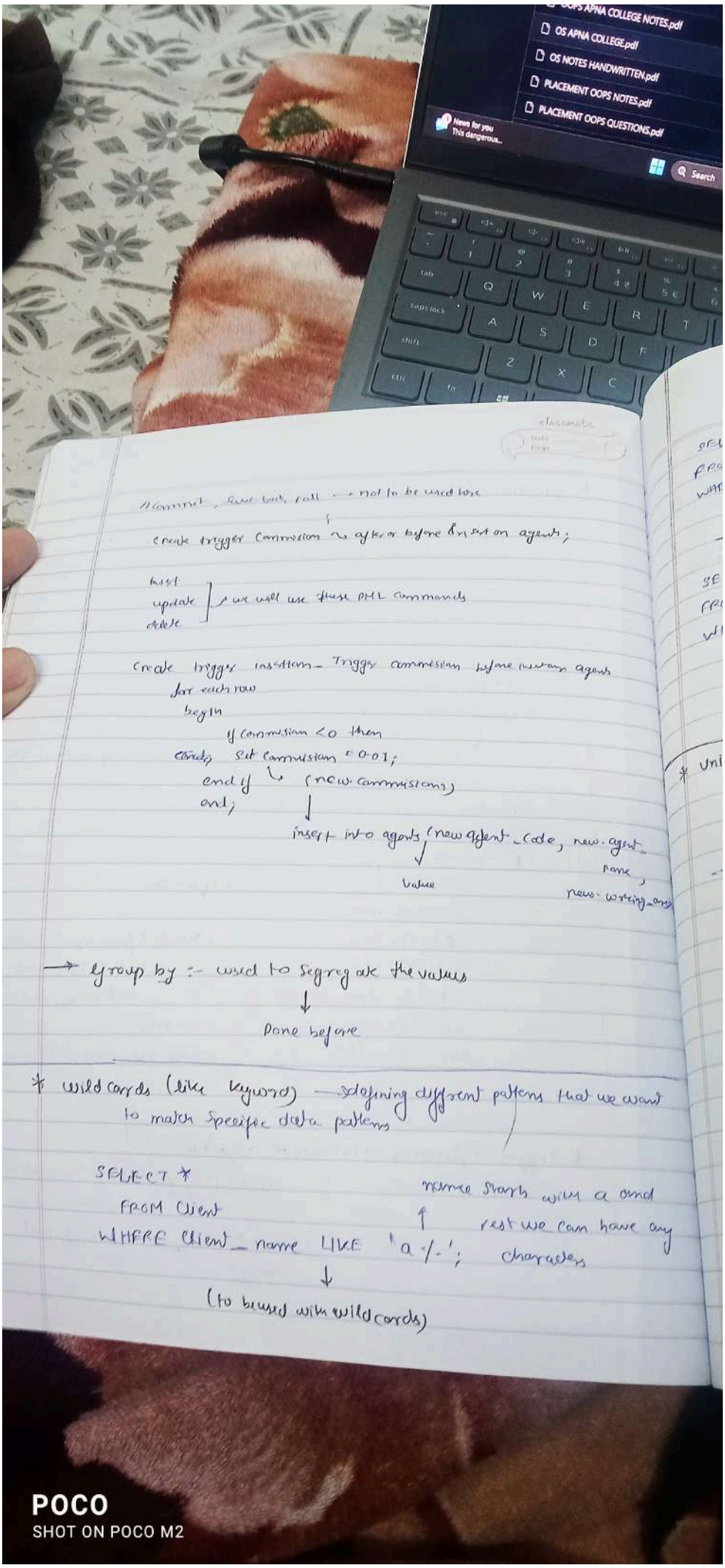
Any condition
↓
print all

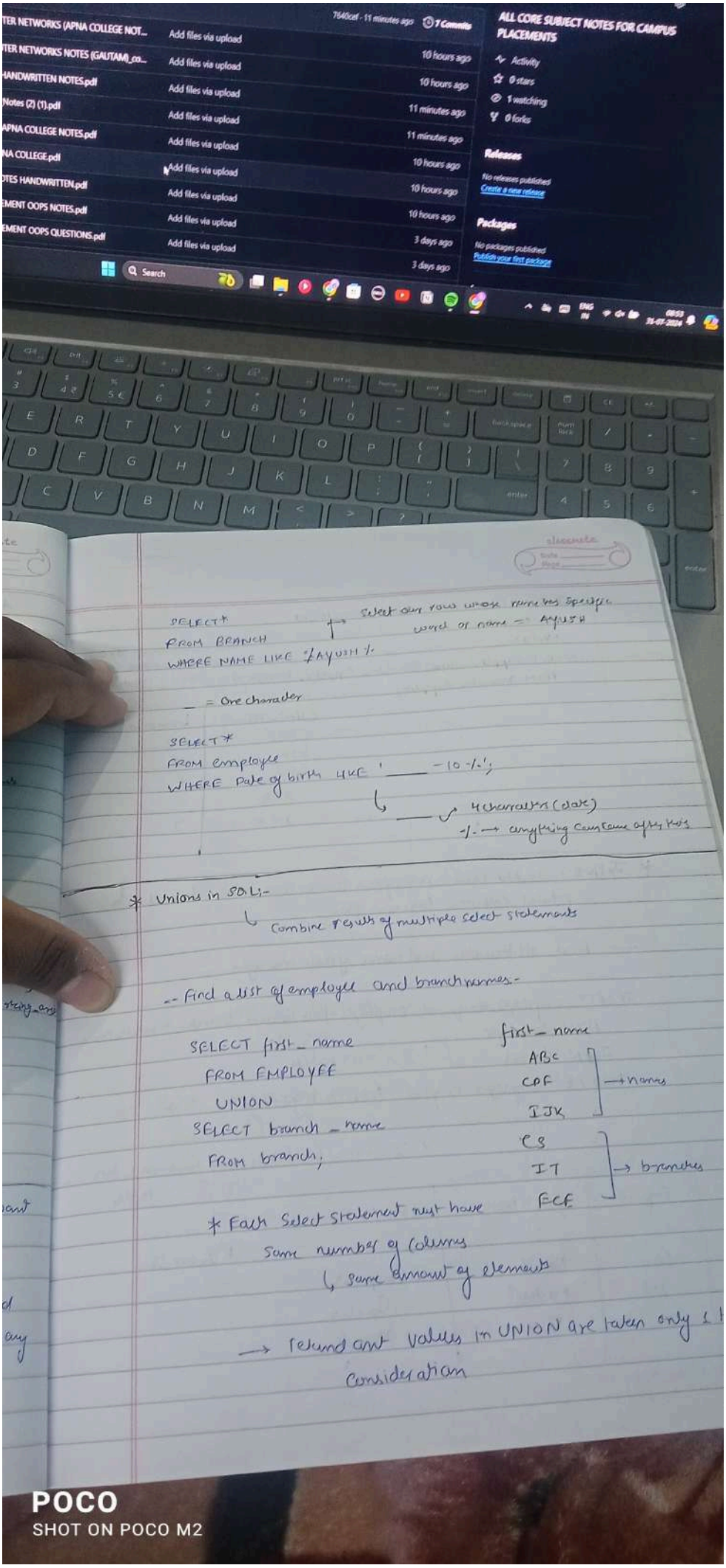
* ALL and ANY → logical operators in SQL
↓
read once from tutorials
↓
Mainly not used but for references, Study

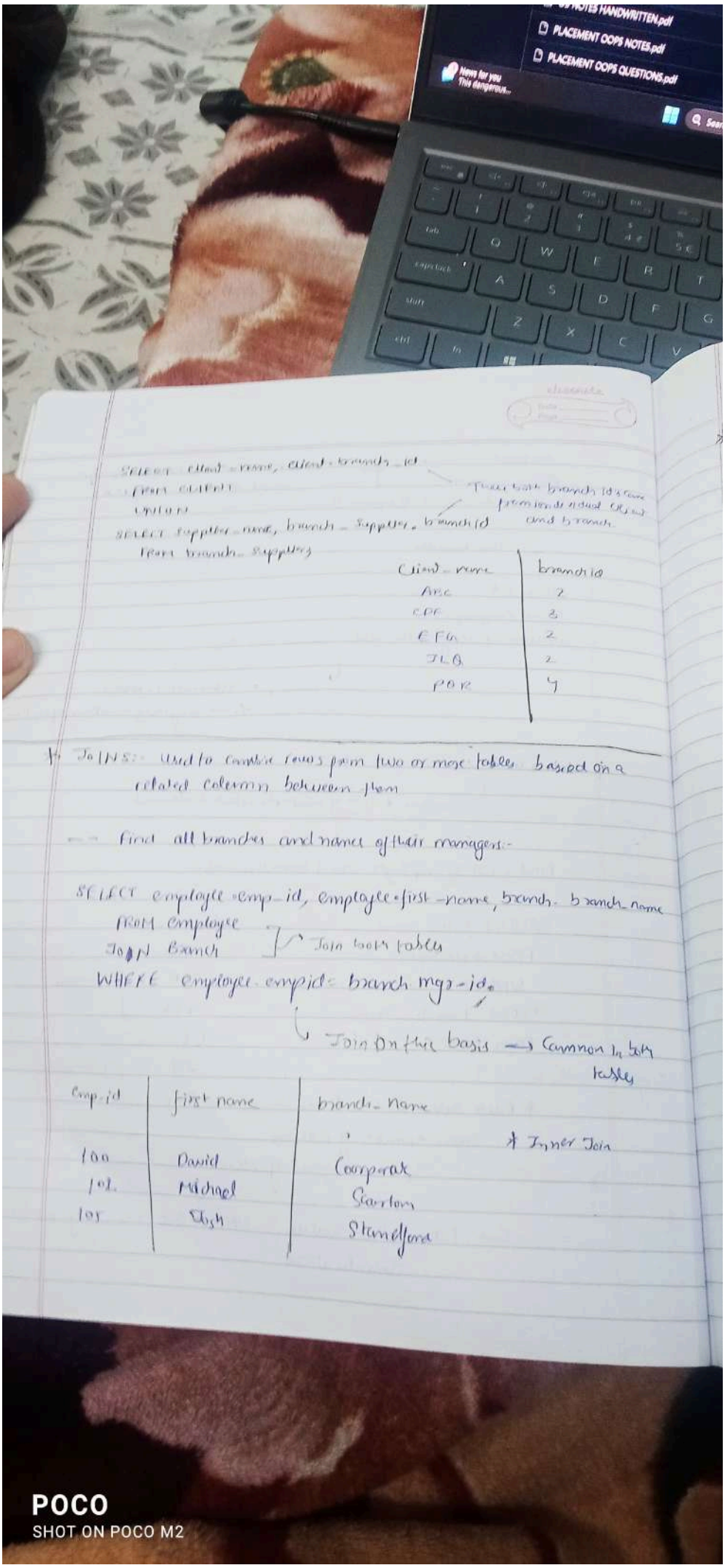
→ Subquery → all condition must be true

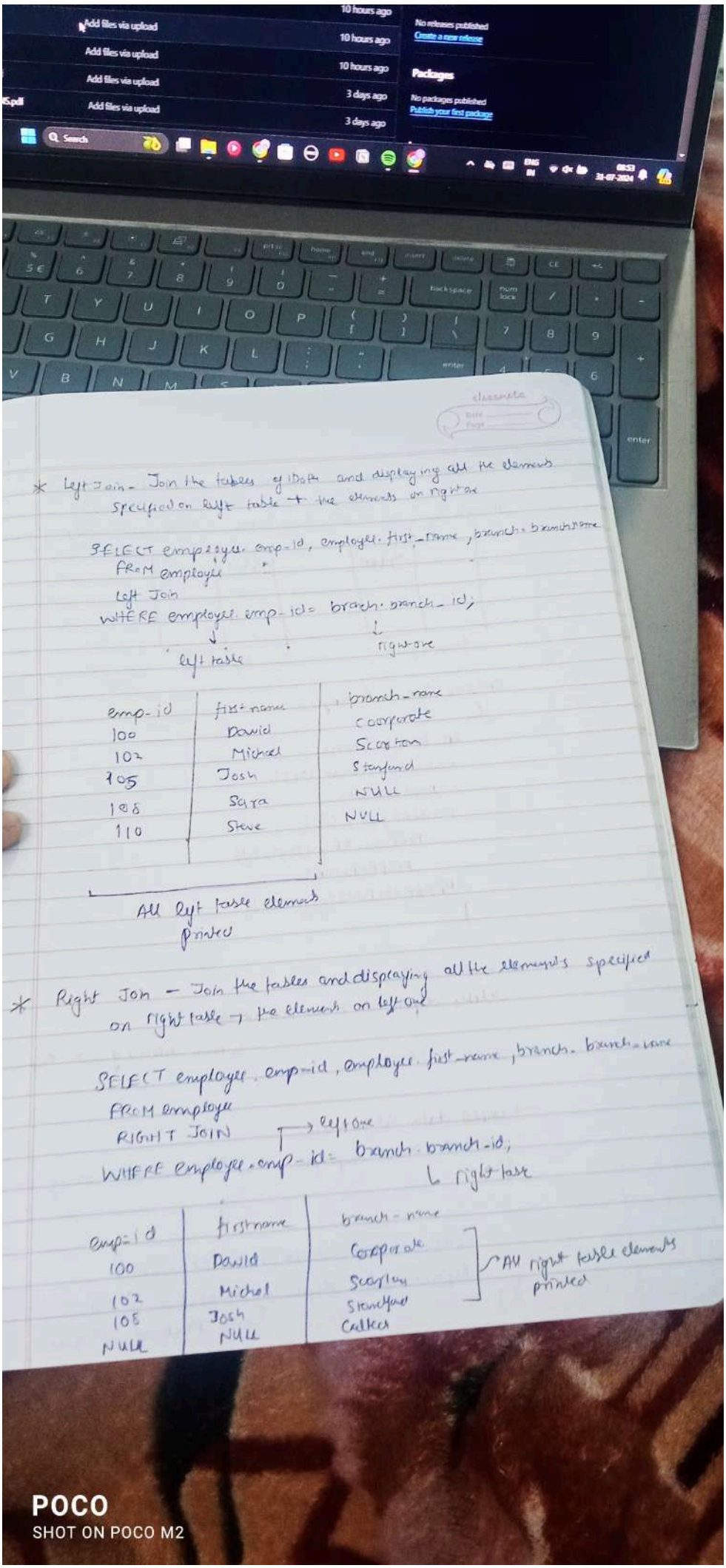
Select * from
customers
where customer_id = all (select customer_id from orders where amount >





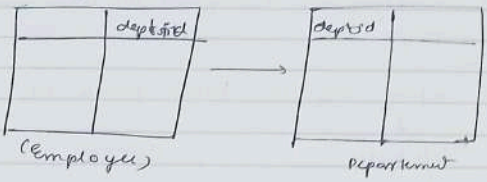






* on delete :- For this we need to understand foreign key

Foreign key – key to link two tables (To maintain integrity of data)




```
CREATE TABLE Employee
(
    ID int NOT NULL,
    name varchar (255),
    Dept_id int,
    PRIMARY KEY (ID),
        FOREIGN KEY (Dept_ID)
            REFERENCES
    DEPARTMENT (dept_id),
);
```

* ON DELETE CASCADE -

deletes dependent rows in child table

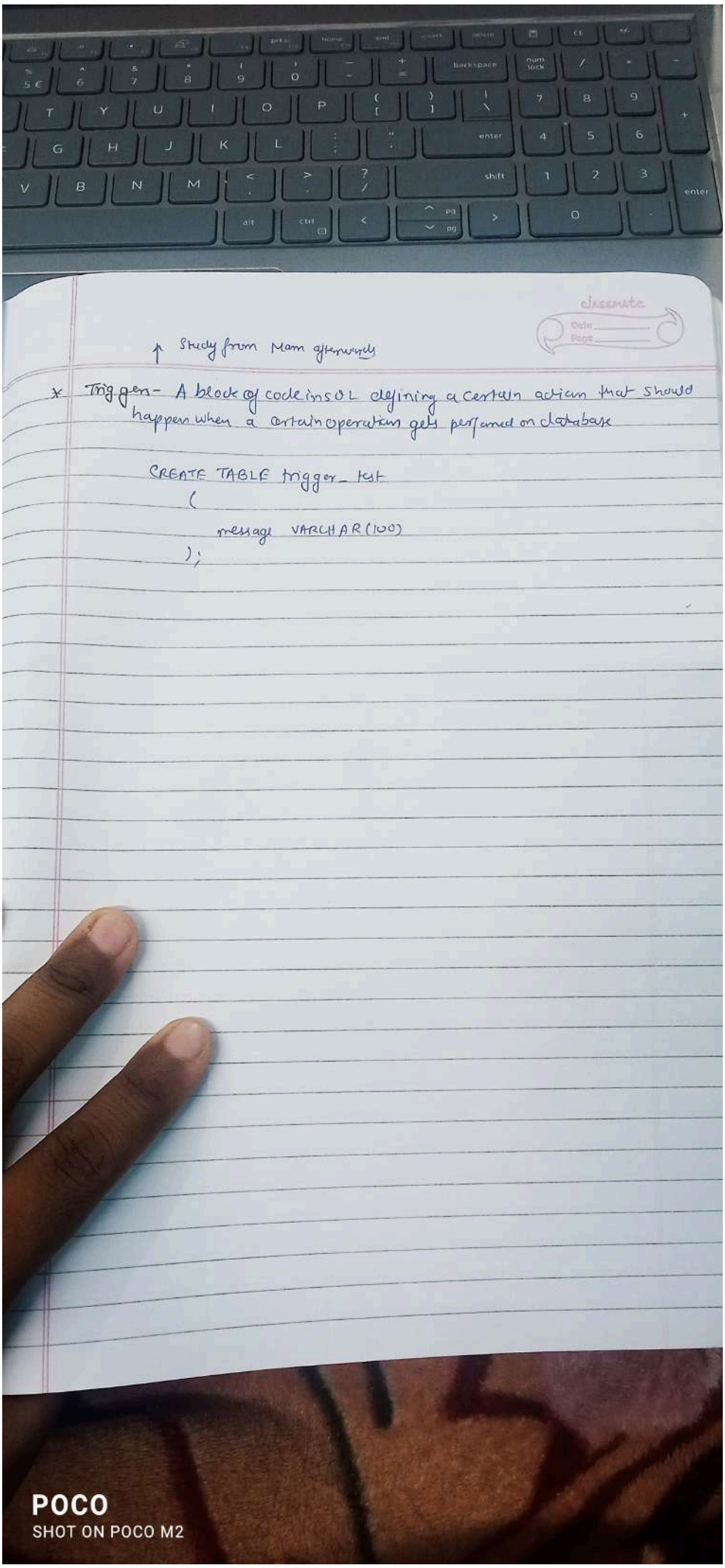
↳ rows deleted of data emp_id

* ON DELETE SETNULL -

→ child data set to NULL

```
CREATE TABLE employee
{
    FOREIGN key (dept_id) REFERENCES
    Department (dept_id) ON DELETE CASCADE
                         ↳ ON DELETE SETNULL
};
```

SETNULL → setting values to NULL



↑ Study from Mam afterwards

* Triggers - A block of code in SQL defining a certain action that should happen when a certain operation gets performed on database

```
CREATE TABLE trigger_test
(
    message VARCHAR(100)
);
```

→ student_id	int(12)	PRI
name	varchar(20)	
majors	varchar(20)	

→ To delete the whole table — DROP TABLE (TABLE_Name)

↓

for this case we have STUDENT

→ To modify our table

suppose we have already created a table and we need to modify it (for eg - inserting a new column)

↓

ALTER TABLE STUDENT ADD CGPA DECIMAL(3,2);

↓

(added a column of CGPA in our existing table)

Added

Student_id	Name	Majors	CGPA

- total number of digits
- digits after decimal place

→ To remove a column from table

↳ ALTER TABLE STUDENT DROP COLUMN CGPA;

↓

(CGPA column removed from table)

Student_id	Name	Majors

↳ CGPA removed