

SQAL → Structured english query language
↑
SQL (Structured query languages)

* SQL is a language used to interact with relational database management systems (RDBMS)
↳ Software application used to create and manage different databases

all DB tables
↓
relational DB
* Database Schema - Basically all of the database tables and relations one collectively stored in a database.

(Database) → Any collection of related information (can be stored in different ways)

↳ Database
↓
Stored info
eg phone book, shopping list, Todo list, Facebook's user base etc
(Stores phone numbers of people)

→ Database can be stored in different ways → • on paper • on a computer
• In mind • Comments
etc. - etc. -

* Databases can be present in various environments

↳ But its usage case, like Amazon database stores trillions of data whereas a list would store 10-20 data members

* DBMS → Database management Systems → special software program that helps users create and maintain a database.

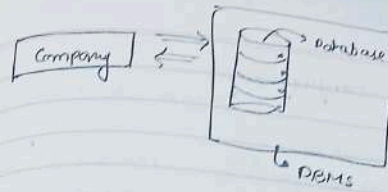
→ Makes it easy to maintain large amount of information

→ Handles Security

→ Backups → Importing / Exporting data

→ Interacts with Software applications

↳ programs to work in DBMS - SQL



* Company for eg. Amazon will interact with DBMS 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)

• Organizes data into one or more tables

* A table has columns & rows

* A unique key identifies each row

→ rows = tuples, columns = attributes

Non-Relational Databases (NoSQL)

• Organizes data in anything but a traditional table → not a relational database

→ Key-value stores → graphs

• Documents (JSON, XML etc)

• Graphs, flexible tables

↳ Storing of data in these databases

Eg (Relational Database) → MySQL, PostgreSQL etc

(STUDENT TABLE)

* ID#	Name	Major
1	Jake	Biology
2	Kate	Sociology
3	Clare	English
4	John	Chemistry

→ Here ID is acting as a

key uniquely identifies each row in our table

↳ ID has unique values for every row

during → insertion
→ check condition

stu-age intcheck (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.)

↓
(Relational database management System)

→ used to define tables and structures

↓
Helps us to create and maintain relational database

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

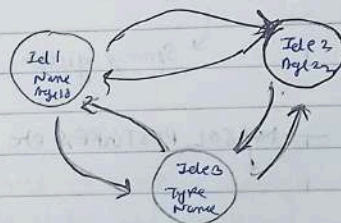
* Non-Relational Databases (NoSQL) → uses other data structures

↓
• 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
"lmno"	etc...

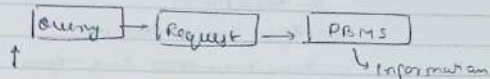
* Like RDBMS, we can have non-relational database management system (NRDBMS)

↓
Create and maintain non-relational database
→ MongoDB, firebase etc.

* Query - Relational database falls under this category, most language standard

* Most DBMS will implement their own language for performing CRUD and administrative operation on database

* Database Query -



Queries are request made to DBMS for specific information

→ As database structure becomes more complex, difficulty to get specific pieces of information we want becomes high.

↓
for eg - Google Search is a query

* TABLES and KEYS :-

(STUDENT)

→ Tells information of STUDENTS

representing attributes 6 rows	STUDENTID	NAME	MAJOR
	1	Jack	Biology
	2	Kate	English
	3	Clare	Sociology
	4	Jack	English
	5	Mike	Comp Science
	6	Vivian	Biology

Column

↓
defines a single attribute

↳ (name, major)

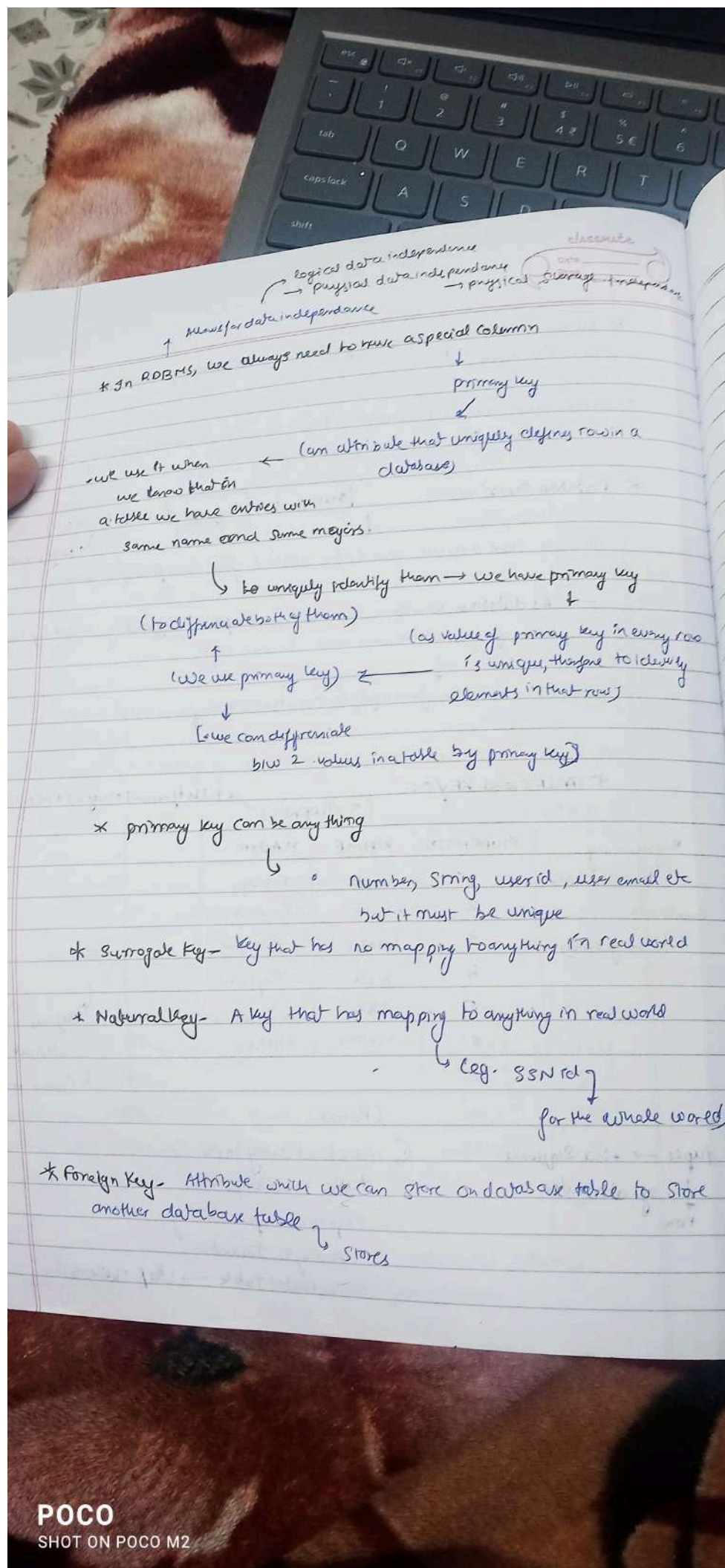
(Rows)

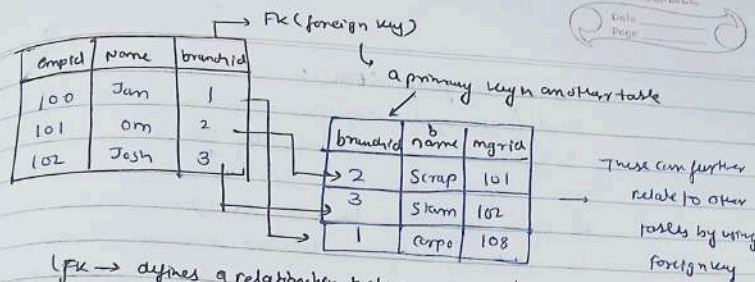
↳ individual entry in the table → a single row

represents information of a single student in

Student Table → (1, Jack, Biol)

tuple → is a sequence of attributes
↓
row





(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 element

* Composite key - Any key with more than 1 attribute

	<u>branch id</u>	<u>Supplier name</u>
They both	2	Hammer Mill
together can	2	Uniball
uniquely identify	2	Patriot paper
each row	3	Uniball
	3	Hammer Mill

* 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.

* Data Query Language (DQL):-

- used to query database for information
- get information that is already stored there

* Data Definition Language (DDL):-

- used for defining database schema
- ↳ Create, ~~delete~~, Drop, Alter, Rename

* Data Control Language (DCL):-

- ↳ Insert, update, delete
- used for controlling access to data in database
- user and permissions management

GRANT
REVOKE

* Data Manipulation Language (DML):-

used to insert, update and delete data from data base

* TCL (Transaction Control Language) → start transaction, commit, roll back etc.

* Queries:-

A query is a set of instructions given to RDBMS that tell the relational database what information you want it to retrieve for you

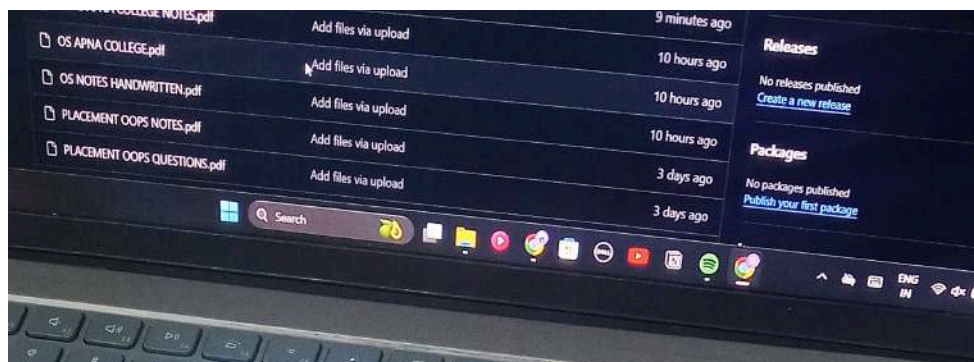
→ Things of data in a DB (database)

→ often hidden in complex schema

→ Goal is to only get the data you need

```
SELECT employee_name, employee_age  
FROM employee  
WHERE employee_salary > 30,000;
```

Create database giraffe - SQL command given to server to
Create a database named giraffe



Different types of data that we can store in our database:-

1. INT --- whole numbers
2. decimal (M,N) --- decimal numbers - exact value
3. Varchar (1) --- string of text of length 1, if (100) → length 100
4. BLOB --- Binary large object, stores large data → I/O store images etc
5. DATE 'YYYY' - MM-DD
6. TIME STAMP "YYYY-MM-DD", HH:MM:SS " → used to record time

→ To create table:-

CREATE TABLE Student ();

(these both are SQL reserved words)

CREATE TABLE STUDENT (student_id INT PRIMARY KEY, name VARCHAR(100), major VARCHAR(20);

→ STUDENT table has 3 columns → STUDENT-ID, NAME, MAJOR

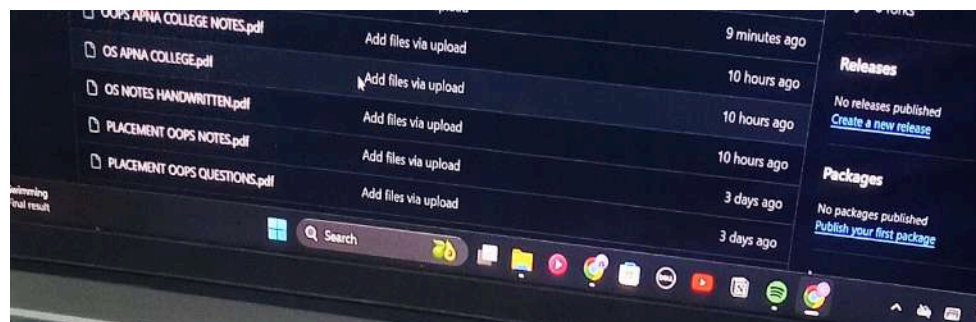
(attributes) (stores integer) (stores string of max length 100) (stores string of max length 20)

student id	name	major
1	Jack	CSE
2	Daniel	IT
3	Ava	ECE

→ To uniquely identify each row, we use this as our primary key

→ To define primary key differently - PRIMARY KEY (STUDENT-ID)

→ DESCRIBE student; → describes our student table initially created



→ describe table → desc student;
(table - name)

Ad insertion:-

→ To insert a piece of information into a table:-

INSERT INTO STUDENT VALUES (gauram, gauram, CSE);
(STUDENT) (table name)

(insert data in a way we setup initially in our table
ie - ID, NAME, MAJOR
↓ ↓ ↓
1 gauram CSE)

Query for our table

ID	NAME	MAJOR
1	Raj	ECE
2	Jack	CSE
3	Nick	MECH

```
CREATE TABLE STUDENT (ID INT PRIMARY KEY,
NAME VARCHAR(25),
MAJOR VARCHAR(25));
INSERT INTO STUDENT VALUES (1, 'Raj', 'ECE');
INSERT INTO STUDENT VALUES (2, 'Jack', 'CSE');
INSERT INTO STUDENT VALUES (3, 'Nick', 'MECH');
SELECT * FROM STUDENT;
```

print table after inserting values init

→ By any chance if we are missing any info of a row and we need to specify it as NULL, for eg → we don't have major for a
ID=3.

∴ ~~INSERT~~ INTO STUDENT (ID, NAME) ~~VALUES~~ VALUES (3, 'Nick')
INSERT
→ Major → NULL

* if a column value cannot be null → for eg NAME cannot be NULL.

∴ name VARCHAR(20) NOT NULL;
↳ (cannot be null)

UNIQUE → value of this column must have unique values for each row.

major VARCHAR(20) UNIQUE;
↳ no duplicate entry.

→ UNDECIDED:-

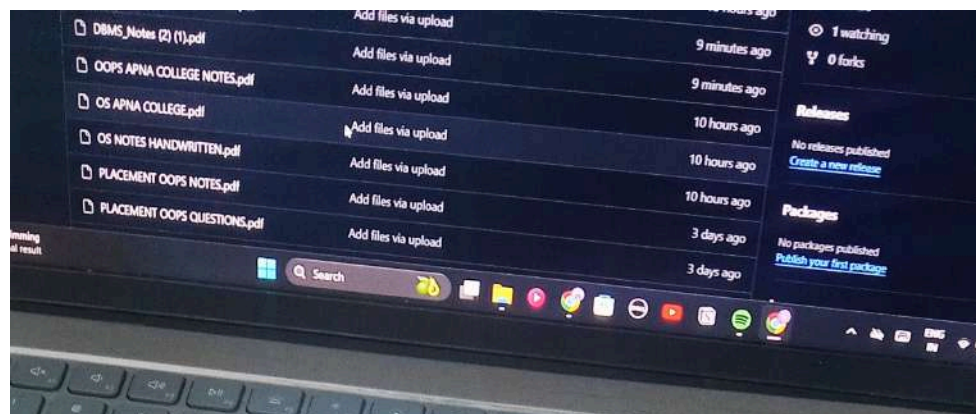
↳ if we don't know the value of a particular field
↓
we could replace that column value with UNDECIDED

for eg. major VARCHAR(20) DEFAULT 'UNDECIDED';
INSERT INTO ~~TABLES~~ (Student-id, name) values (1, 'Raj');

if we didn't add our major
↳ value would be assigned UNDECIDED by default.

output →

Student-id	Name	Major
1	Raj	Undecided



Auto-Increment

instead of giving a particular value, we would get assigned values again and again. We can use AUTO-INCREMENT.

as per no. of rows in table

Student id	INT AUTO INCREMENT	Student id
	1	1
	2	2
	3	3
	4	4

(as per no. of inputs)

INSERT INTO Student (name, major) VALUES ('Jack', 'Biology');
(Student id = 1) Automatically

INSERT INTO Student (name, major) VALUES ('KATE', 'MATHS');
(Student id = 2) Automatically

Update and Deletion of Rows - (inside database table)

Update

we need to update the value in a row, for eg. we need to update our major = 'Biology' to major = 'Bio'.

↓

(Table name)

UPDATE Student;
SET MAJOR = 'Bio';
WHERE major = 'Biology';

→ replace all major = 'Biology' to major = 'Bio' in every row where major = 'Biology' was present.

→ we can update a row's value without changing the value 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.

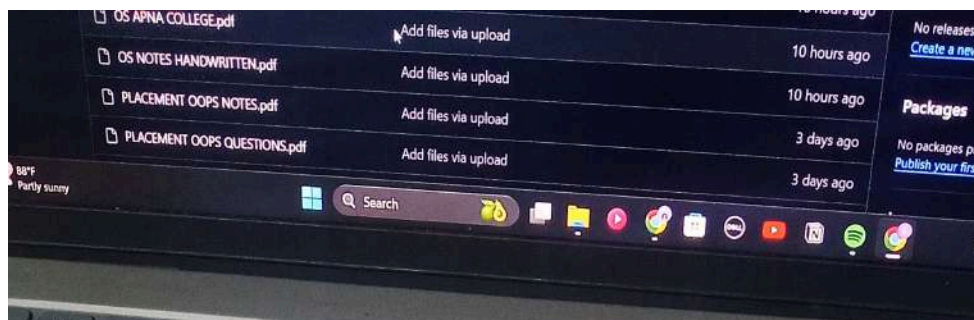
for eg Major 1 = 'Bio'; Major 3 = 'Biochem';
 Major 2 = 'Chemistry'

∴ 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



classmate
Date _____
Page _____

If we remove where statement, changes would be affected to each row.

UPDATE STUDENT;
SET MAJOR = 'UNDECIDED';

every major in every row changed to Undecided.

Deletion In Table:-

↳ To delete a specific row from our table

↓

this delete command is based on (table's name) used

→ DELETE FROM STUDENT
where name = 'Vivek';

↓
or

where name = 'Vivek' and major = 'cs';

↓
or where student id = 13;

↓
(row deleted)

→ If we remove our where statement

↳ all rows would be deleted from table.

POCO
SHOT ON POCO M2

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

SELECT → keyword to interact with DBMS 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:-

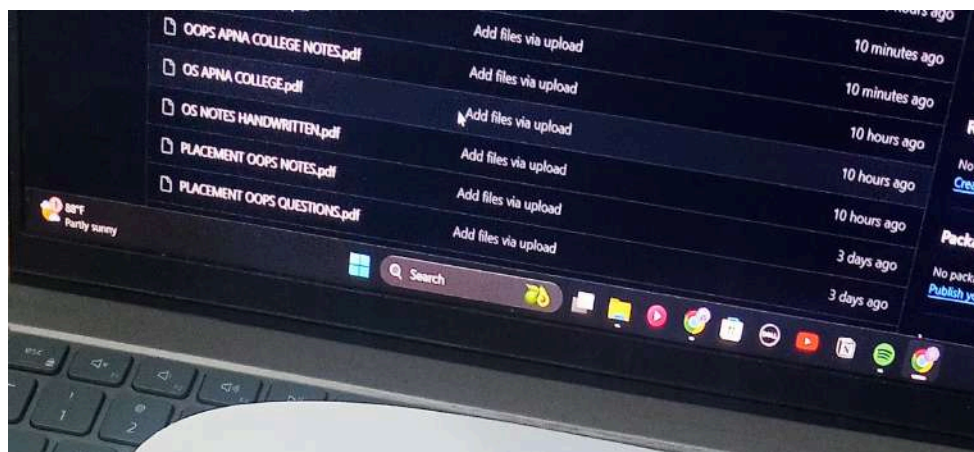
we can order (ascending or descending) our column's information

↓
ORDER BY NAME;

(Ascending) { (Alphabetical) } → Arranges name from ascending order

ORDER BY STUDENT-ID

1
2
3
4 } order



execute

ORDER BY NAME DESC;
 ↓
 order alphabetically in descending order

order by student_id DESC;
 ↓
 10-
 9-
 8-
 (Descending order)

→ Sorting sub columns:-

```
SELECT *
FROM STUDENT
ORDER BY major, student_id;
```

↓
 first order by major → if there are 2 common majors
 orders by student_id

Student id	NAME	MAJOR
1	Jack	Bio
4	Jack	Bio
3	Clare	Chem.
5	Mike	CS
2	Kate	Social

LIMIT → to only extract a amount of rows
 ↳ LIMIT 2 → extracts first 2 rows of table only

POCO
 SHOT ON POCO M2

* WHERE → keyword to select an information from table

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

select and print names and major accordingly

name	major
Jack	Biology
Clare	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

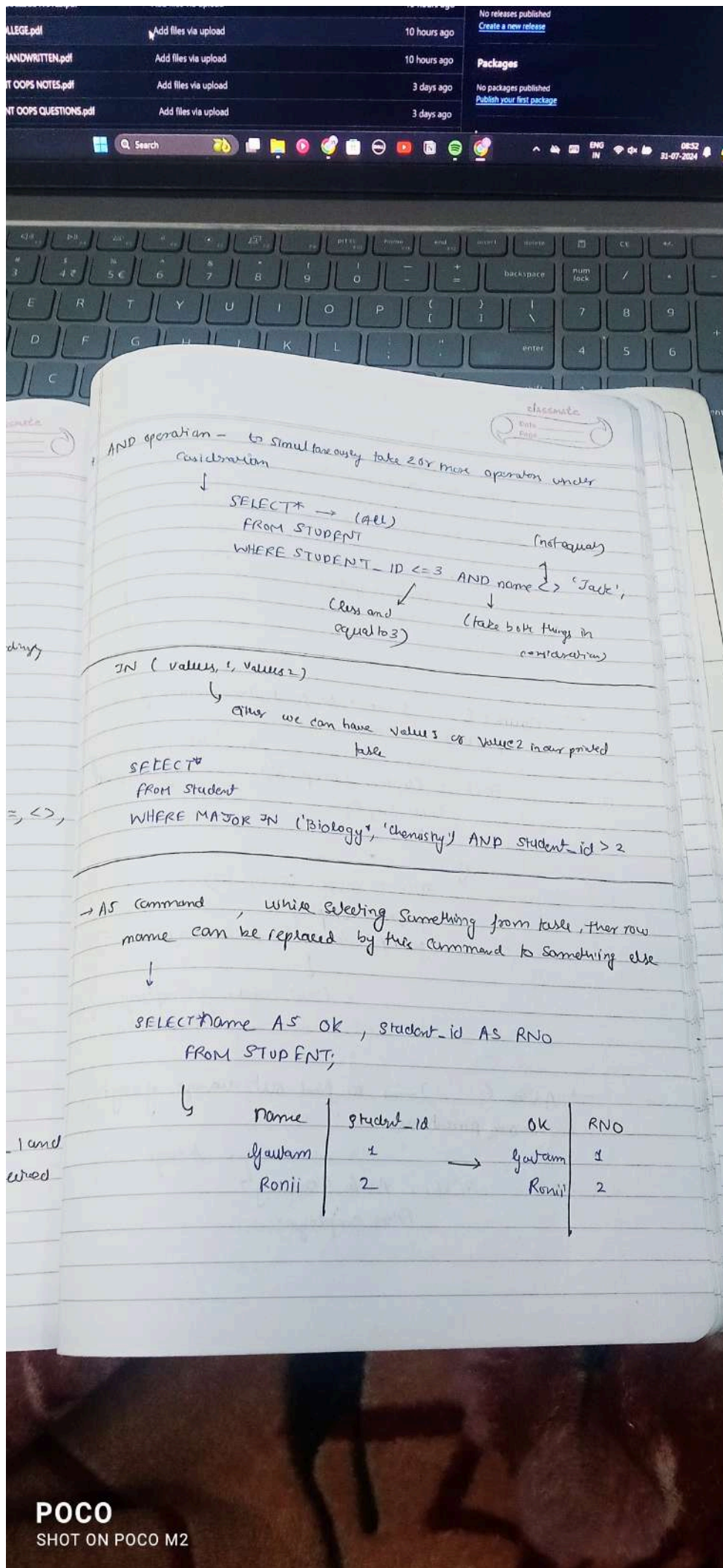
or $<=>$ less than

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

StudentId 1 and

2 selected

↓
their names



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

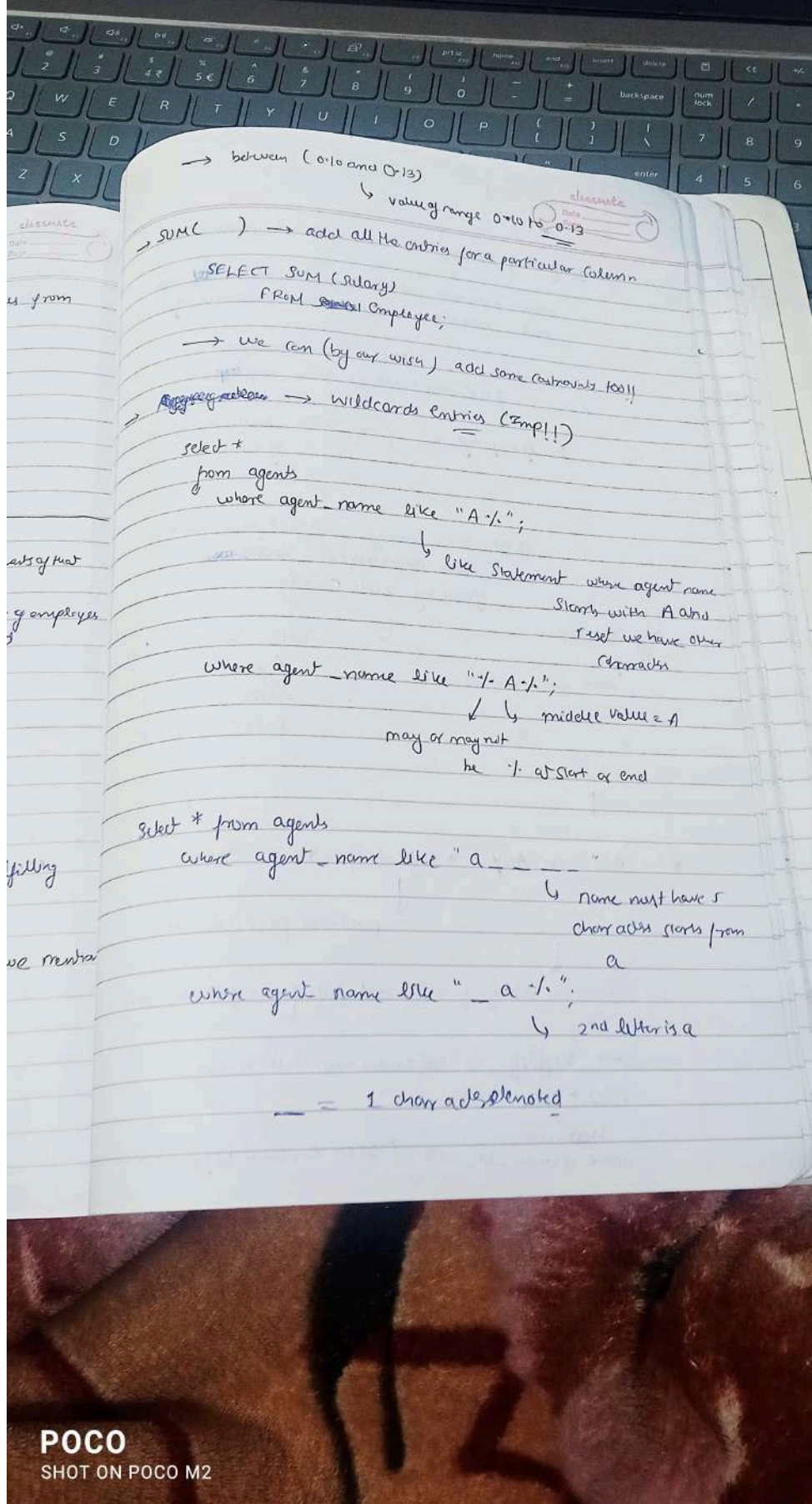
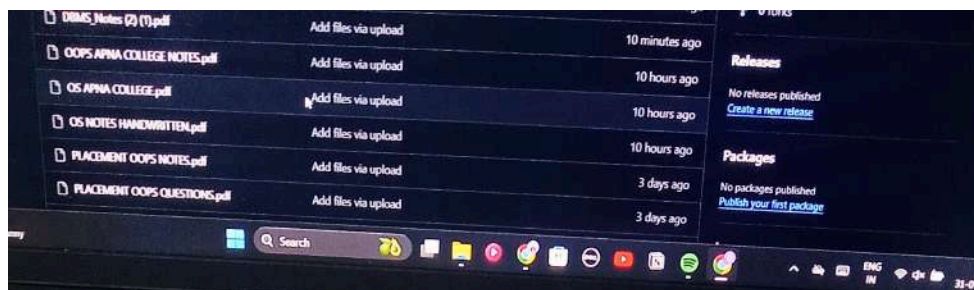
`SELECT COUNT (emp-id)
FROM employee;` → Count the number of employees by emp-id

↳ `SELECT COUNT (emp-id)
FROM employee
WHERE sex = 'F' AND BRANCH = 'IT';`

↓
(Count number of employees fulfilling this criteria)

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

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



* aggregate functions → min and max function

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

~~where cust-country = 'India'~~

group by cust-country;

Any condition

↓
print all

cust-country	count(*)
Australia	2
India	4
Pakistan	5

* ALL and ANY → logical operators in SQL

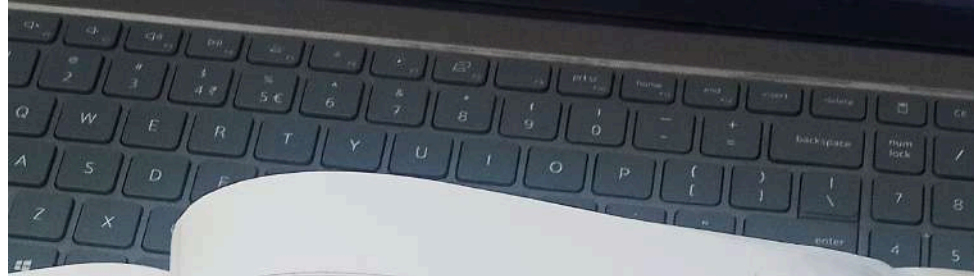
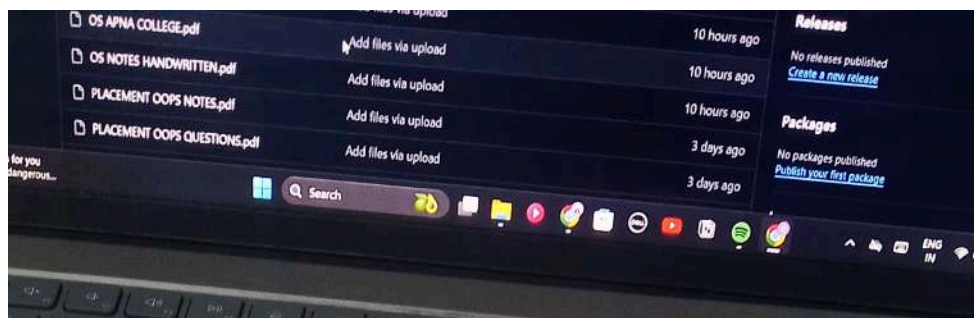
↓
read once from tutorials

↓
Majory not used but for reference study

→ Subquery → all condition must be true

select + from
customer

where customer-id = all (select customer id from orders where amount >



UNION (Joins \rightarrow left join, right join)
 \downarrow
 full join = left + right join
 (natural join) and inner join

Cartesian product:-
 select * from agents
 inner join customer; \rightarrow product
 \downarrow
 every table \rightarrow every row combines with all rows of another table

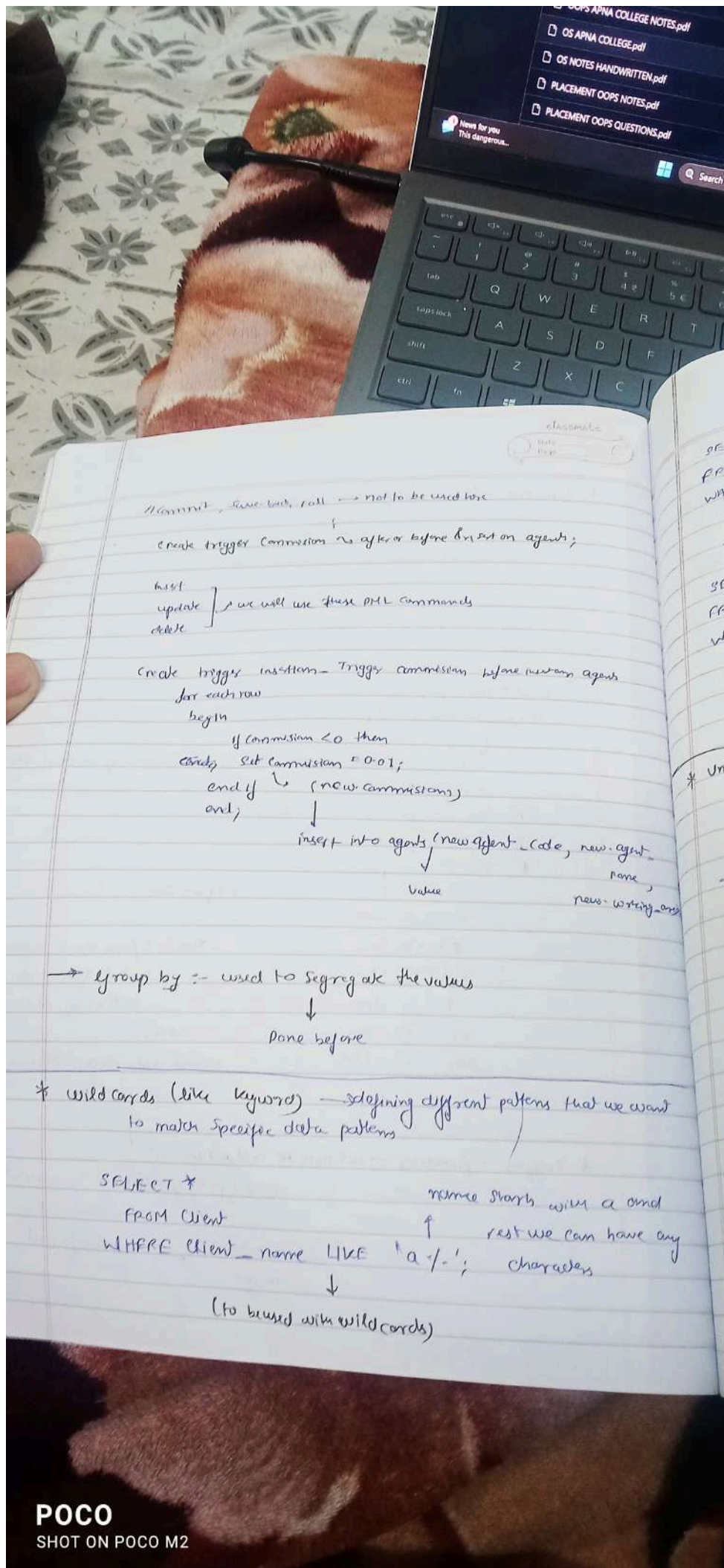
P ₁ 100	and	1 a
P ₂ 300		2 b
P ₄ 500		

\downarrow

1 a	P ₁ 100	* Equi-Join
2 a	P ₂ 300	
1 a	P ₄ 500	
2 b	P ₁ 100	
2 b	P ₂ 300	
2 b	P ₄ 500	

select * from agents, customer
 where agent-AGENTSCOPE
 = customer-AGENTSCOPE
 and
 where cust_name = 'Michael';

* Triggers \rightarrow functions \rightarrow we need to call or run
 (a type of procedure \rightarrow we directly
 don't call or
 would automatically modify that
 trigger)



commit, save, back, roll → not to be used here

create trigger commission → after or before insert on agents;

insert
update
delete } we will use these DML commands

create trigger insertion - Trigger commission before insert on agents
begin

if commission < 0 then
commit; set commission = 0.01;
end if
end;

insert into agents (new agent code, new agent
value none,
new writing order)

→ group by :- used to segregate the values

↓
done before

* wild cards (like keyword) - defining different patterns that we want
to match specific data patterns

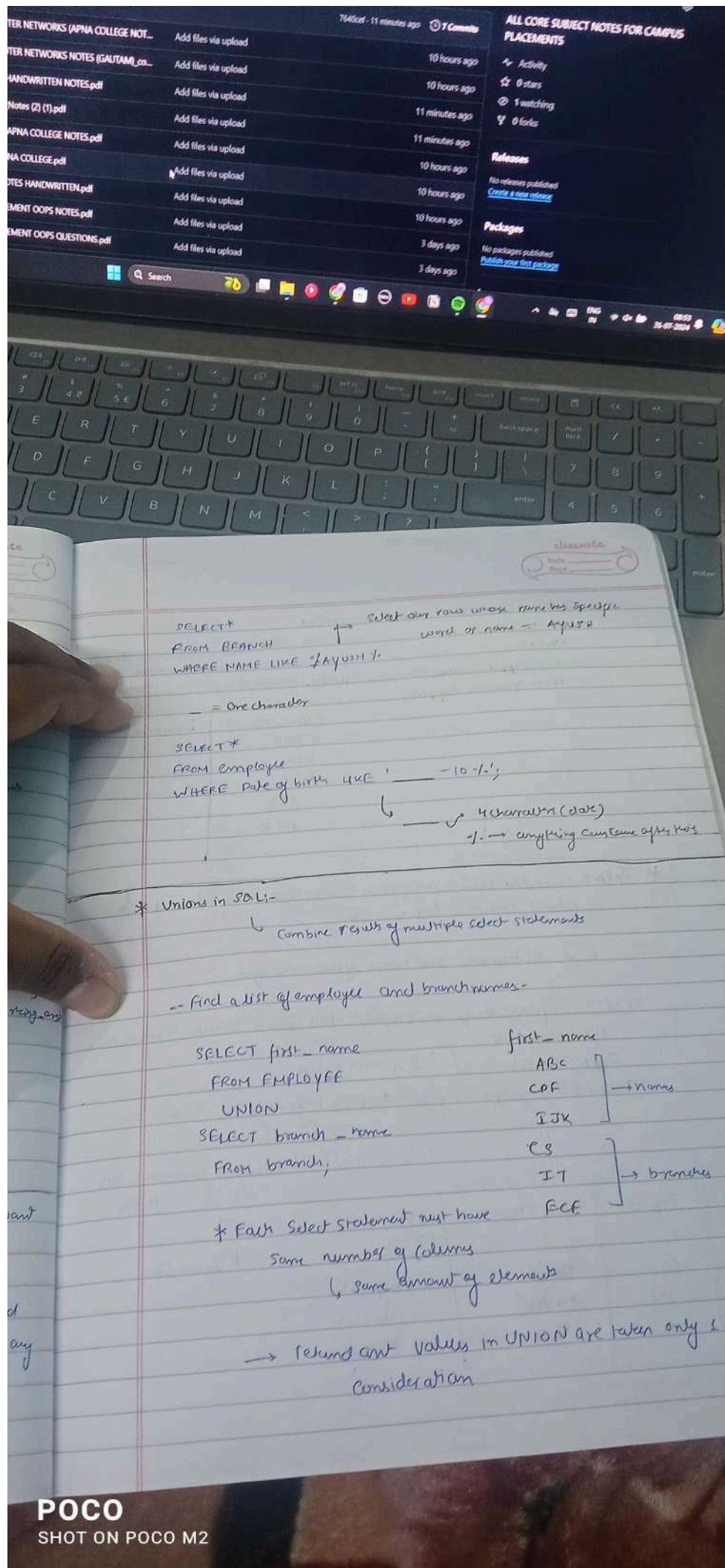
SELECT *

FROM client

WHERE client_name LIKE 'a/-';

name starts with a and
rest we can have any
characters

↓
(to be used with wildcards)



```

SELECT client_name, client_branch_id
FROM CLIENT
OPTION
SELECT supplier_name, branch_supplier_branch_id
FROM branches_supplier

```

These both branch id's are
primary id of client
and branch.

Client_name	branch_id
ABC	2
CDE	3
EFG	2
ILG	2
POK	4

* **JOINS**:- Used to combine rows from two or more tables based on a related column between them

Find all branches and name of their managers:-

```

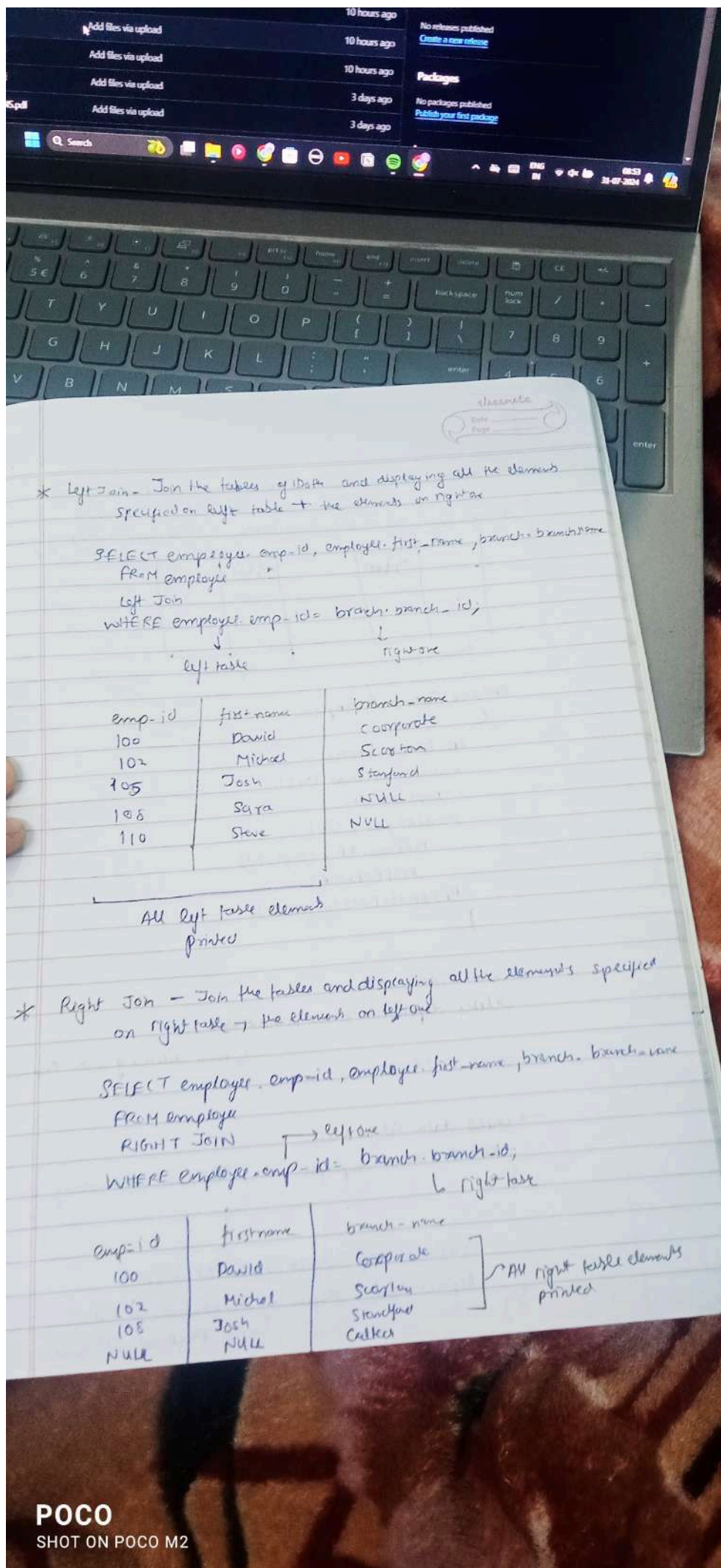
SELECT employee.emp_id, employee.first_name, branch.branch_name
FROM employee
JOIN branch
WHERE employee.emp_id = branch.mgr_id

```

Join on the basis → Common in both
tables

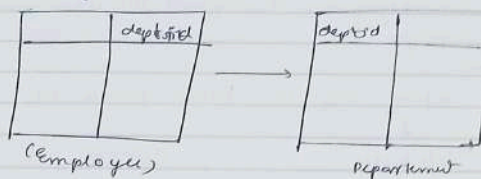
Emp_id	first name	branch_name
100	David	Corporate
102	Michael	Stanton
105	Josh	Stamellford

* Inner Join



* 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 SET NULL -

→ child data set to NULL

```
CREATE TABLE employee
{
```

SET NULL → setting value to NULL

```
FOREIGN KEY (dept_id) REFERENCES
```

```
department (dept_id) ON DELETE CASCADE
```

```
ON DELETE SET NULL
```


↑ Study from Mam ghanwani

classmate
Date _____
Page _____

* Trigger - 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(11)	PRI
name	Varchar(20)	
major	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 my
existing table)

Student_id	Name	Major	CGPA

total number of
digits

digits after
decimal place

→ To remove a column from table

↓
`ALTER TABLE STUDENT DROP COLUMN CGPA;`

Student_id	Name	Major

↓
(CGPA column removed from table)

CGPA removed