ERD:

* Entity – nouns
  + Attributes are also noun that we can assign a value
  + can the value of this attribute or set of attributes be repeated
    - if no then they are candidate keys. 🡪 ID, ID1,….
* Relations – verbs
  + Normal/ Relation Type
  + ISA (do not put ID)
  + Weak Entity (always one to many) (do not put ID)
  + Association class/ linked attribute
  + Ternary relation

CONVERT ERD to table

* Normal Entity
  + Create table statement and make sure one of IDS become primary key and other

ID are unique. If you have ID2,ID2 it means you need a composite key

e.g F\_name ID1, M\_NAME ID1, L\_name ID1

UNIQUE(F\_name, M\_NAME, L\_name)

* ISA (Copy primary key of parent and make sure you show it’s a foreign key)
* Weak Entity ( primary key is a composite key of the pk of main entity, the attribute in the box attached to the main entity , make sure you show it’s a foreign key)
* Many-to-Many, linked attribute, association class (always has its own table, copy the PK of all tables attached, primary key becomes a composite key of all the PKs. Do not forget to show the foreign keys)
* One to Many
  + ONLY copy the PK from one side to many side and make it a foreign key
  + DO NOT CHANGE PK
* One to One
  + Copy PK of one of the entities to the other and make it a foreign key
  + DO NOT CHANGE PK
* Constraints
  + Primary key
  + Foreign key
    - On delete/ on update (No ACTION, SET NULL, check syntax in lecture note)
      * For weak entity and ISA relation always say on delete cascade on update cascade
    - Deferable/ Immediate (check syntax in lecture note)
  + Unique (all the IDS we did not choose to be primary key, e.i candidate keys)
  + Default value
  + Not null
  + Check: is restriction on the value that an attribute can take
  + Assertion it’s like a check bur requires multiple table so we create it separately

City can be dubai abudhabi or ajman

City ENUM(‘Dubai’,Abudhabi’, ‘AJMAN’)

City char(100) (check city in (‘Dubai’,Abudhabi’, ‘AJMAN’))

employee

Salary can be between 5 to 10 K

Salary int (check salary between 5000 to 10000)

Department

Salary\_min

Salary\_max

Each employee should be paid between the department salary range and less than manager

Create assertion as

Query

(NOT USING JOIN)

SELECT column\_names\_separated\_by\_comma

FROM table\_names\_separated\_by\_comma

WHERE condition [AND/OR more\_conditions)

GROUP BY (column\_name)

HAVING condition [AND/OR more\_conditions)

ORDER BY clumn\_name ordering\_type(ASC or DESC)

LIMIT n (don’t worry abt this one)

condition format is

column\_name operator value

operators:

>, >=, <, <=, =, LIKE (for strings), IS/IS NOT (for null values)

WHERE filters the rows of the tables

HAVING filters the groups created by group by statement

aggregate function: SUM(column), AVG(column), MIN(column), MAX(column) COUNT(column)

* Basic

SELECT \* FROM employee;

SELECT fname, lname, salary from employee;

Select \* from employee WHERE city=’Dubai’

Select \* from employee WHERE city like ’D%’ – start with D

Select \* from employee WHERE city like ’%D’ – ends with D

Select \* from employee WHERE city like ’%D%’ – D can be anywhere

Select \* from employee WHERE city like ’\_D%’ single letter before D and any number of letters after it

Select \* from employee WHERE city=’Dubai’ OR city= ‘Ajman’ OR city= ‘Abudhabi’

Select \* from employee where city in (’Dubai’, Ajman’,‘Abudhabi’)

Select \* from employee where city in (’Dubai’, Ajman’,‘Abudhabi’) AND salary>=10000

Select \* from employee where city is null

Basic with aggregate function

Select min(salary) from employee where department=’IT’

Select distinct(department) from projects;

Basic with change in columns

Select price\*0.9 AS new\_price from products. --- give 10% discount on price

* Nested query basic

select i\_salary from instructor;

/\*show the difference between

average salary and the salary each person is paid\*/

select salary-(select avg(salary) from instructor)

from instructor;

/\*show all instructors who are paid the minim sallary\*/

select \* from instructor where i\_salary=(

select min(i\_salary) from instructor);

/\*show email from the instrctors who are not working \*/

SELECT i\_email from instructor

WHERE i\_email not in (select i\_email from schedules);

SELECT i\_email from instructor

WHERE NoT EXISTS (select \* from schedules

where schedules.i\_email=instructor.i\_email);

/\*get all male employees who are being paid higher than

any other female employees\*/

select \* from instructor where

i\_gender='M' and i\_salary>all(select i\_salary from

intructor where i\_gender='F')

* Multiple tables

Option 1: list all tables, separate by comma, add join condition

SELECT column\_names\_separated\_by\_comma

FROM table\_names\_separated\_by\_comma

WHERE tb\_name1.PK=tb\_name2.FK

[AND tb\_name2.PK=tb\_name3.FK]

[AND/OR condition [AND/OR more\_conditions) ]

GROUP BY (column\_name)

HAVING condition [AND/OR more\_conditions)

ORDER BY clumn\_name ordering\_type(ASC or DESC)

LIMIT n (don’t worry abt this one)

Option 2: use JOIN statement

SELECT column\_names\_separated\_by\_comma

FROM table\_name1 JOIN table\_name2 ON tb\_name1.PK=tb\_name2.FK

JOIN table\_name3 ON tb\_name2.PK=tb\_name3.FK

WHERE condition [AND/OR more\_conditions]

GROUP BY (column\_name)

HAVING condition [AND/OR more\_conditions)

ORDER BY clumn\_name ordering\_type(ASC or DESC)

LIMIT n (don’t worry abt this one)

* Group by And Group by with nested query

/\*select categories\*/

select distinct su\_cat from subjects;

select su\_cat from subjects

group by su\_cat;

select distinct su\_id, i\_email from schedules;

select su\_id, i\_email from schedules

group by su\_id, i\_email;

/\*show the email, salary of instructors

along with the subject they are teaching)\*/

select i.i\_email, s.su\_id, i\_salary

FROM instructor i JOIN schedules s

ON i.i\_email=s.i\_email

group by i.i\_email, s.su\_id;

/\*show email, salary and number of sessions \*/

select i.i\_email, i\_salary, count(su\_id)

FROM instructor i JOIN schedules s

ON i.i\_email=s.i\_email

group by i.i\_email;

/\*show email, salary and number of subjects \*/

select i.i\_email, i\_salary, count(distinct su\_id)

FROM instructor i JOIN schedules s

ON i.i\_email=s.i\_email

group by i.i\_email;

/\*show email, number of subjects

for instructors who are

getting paid more than min salary)\*/

select i.i\_email, count(distinct su\_id)

FROM instructor i JOIN schedules s

ON i.i\_email=s.i\_email

WHERE i\_salary>(select min(i\_salary) from instructor)

group by i.i\_email;

select i.i\_email, i\_salary, count(distinct su\_id)

FROM instructor i JOIN schedules s

ON i.i\_email=s.i\_email

group by i.i\_email

HAVING i\_salary>(select min(i\_salary) from instructor);

/\*show email, salary, number of subjects

for instructors who are

getting paid more than min salary), show the ones

that are doing more than one \*/

select i.i\_email, i.i\_salary,count(distinct s.su\_id)

FROM instructor i JOIN schedules s

ON i.i\_email=s.i\_email

WHERE i.i\_salary>(select min(i\_salary) from instructor)

group by i.i\_email

HAVING count(distinct s.su\_id)>1;

/\*show email, salary, number of subjects

for instructors who are

getting paid more than min salary), show the ones

that are doing more than zero subjects

and are paid more than

average salary of each gender\*/

select i.i\_email, i.i\_salary,count(distinct s.su\_id)

FROM instructor i JOIN schedules s

ON i.i\_email=s.i\_email

WHERE i.i\_salary>(select min(i\_salary) from instructor)

group by i.i\_email

HAVING count(distinct s.su\_id)>0 AND

i\_salary>ALL (select avg(i\_salary) FROM instructor

group by i\_gender);

* Sets

/\*UINION/INTERSECT/EXCEPT\*

A(1,2,3,4,5,6)

B(1,2,7,8)

A UNION B ===> everything --> (1,2,3,4,5,6,7,8)

A INTERSECT B ===> common values ==>(1,2)

A EXCEPT B ==> values in A but not in B ==>(3,4,5,6)

B EXCEPT A ==> values in B but not in A ==(7,8)

\*/

/\*show all instructors and all

codes\*/

SELECT i.i\_email, su\_id

FROM instructor i left JOIN schedules s

ON i.i\_email=s.i\_email

UNION

SELECT s.i\_email, su.su\_id

FROM subjects su left JOIN schedules s

ON s.su\_id=su.su\_id;

Delete/ Update