**OVERVIEW – SQL**

-SQL Basics

-SQL Servers

-DBMS SYSTEM

-RDBMS SYSTEM

-Database

-Data types

- Numeric Data type

- Date & Time Data type

- String Data type

|  |  |
| --- | --- |
| **My SQL Main Commands**  - Create database (Create a New Database)  - Show Databse (View Databases)  - Drop Database  - Alter Database (Modify Database)  - Create Tables  - Show tables  - Insert Values  - Drop Table  - Alter table ( For New Column Creation)  - Alter Table Modify  - Alter table Drop ( Drop the Column)  - ALter table Rename (Rename the Table)  - Update Table ( To change the Values)  - Delete Statement | **My SQL General Functions**  - Where  - or  - and  - in  - Not in  - >  - <  - >=  - <=  - <> (Not equal to)  - !  - Count  - Distinct  - Count With Distinct  - Order by Asc  - Order By Desc  - Group By  - Limit  - Desc Limit  - Like (\_%)  - Not like  - Between |
| **MY SQL Calculate Functions**  - Sum  - Average  - Min  - Max  - Count | **My SQL String Functions**  - LCase  - UCase  - Left  - Right  - Concat  - Trim  - Char\_Length  - Mid  - Length |
| **My SQL Date Functions**  - Date ADD  - Datediff  - Timestamp DIff  - Date Format  - Year  - Day  - Month  - Now | **Logical Functions**  - IF  - If with and conditons  - If with or conditions  - Count If |
| **Join Queries**  - Inner Join - matching similar rows  - Left Join - add the infos to left table , null in right table  - Right Join - add the infos to right table , null in left table  - Cross join  - Full Outer Join - union of left and right table  - Case end  - when  - then  - double case with end statement  - case with and statement  - case with or statement  - RDBMS with subqueries  - having clause  - basic joins Trigger (Create table after connections) | **Triggers (SQL Automation)**  TYPES-  1. row level - triggers an event at each row updated , inserted or deleted  2. statement level - triggers an event at at table level for each sql statement  Example - creates a new table by applying the triggers  - trigger creation  - trigger timings  - before insert  - after insert  - before update  - after update  - before delete  - after delete  - Trigger with if statement  - trigger with else if statement |

**RDBMS System**

-Two table connection

-Three table connection

**Procedures concepts (SQL Automation)**

- Procedure creation

- Begins & end

- call Procedures

- Alter procedures

- Declering variables

- Store variables

**Pl SQL - Procedural language Structured Query language**

- Procedures

- Functions

- Loops

- Variables

- Triggers

1st week

|  |  |
| --- | --- |
|  | **SQL statements** |
|  | 1. DDL – data definition language  * Database creation * View database * Drop database * Table creation * Alter header * Drop header alter header add |
|  | 1. DML – data manipulation language  * Insert values * Delete values * Update values |
|  | 1. DCL – data control language  * Grant * Revoke |
|  | 1. DQL – data query language  * General functions * Calculated functions * Logical functions * Date/time * RDMS * Joins |
|  | 1. TCL – transaction control language  * Commit – save point * Roll back ( backward ) |

2nd week

|  |  |
| --- | --- |
| **DATA BASE** | 1. default databases –  * information schema * my SQL * performance schema * own databases      1. CREATION  * Create database name ; – ctrl+enter * Use name ; * Drop database name ;   Show databases ; |
| **TABLE** | -- table creation  create table student\_det (S\_no int, student\_name  varchar(30), marks int, gender varchar(12),  primary key(S\_NO));  describe student\_det;  select \* from student\_det; |
|  | -- data creation  create table emp\_det (emp\_id int, emp\_name varchar(35),  city\_state varchar(45), income int, Ph\_no bigint,  primary key(EMp\_id)); |
|  | -- inserting values  insert into emp\_det values  (101, 'Adhavan', 'chennai', 10000, 7200784563),  (102, 'anand', 'trichy', 25000, 8608914600),  (103, 'abhi', 'madurai', 15000, 8765432198),  (104, 'madhavan', 'salem', 12000, 7345234129); |
|  | * ALTER * Add column - alter table emp\_det add gender varchar(35);   alter table emp\_det add gender varchar(35), add age int;   * Drop column - alter table emp\_det drop gender, drop age; * Modify data type – alter table emp\_det modify col\_name datatype ; * Rename column – alter table emp\_det rename to emp\_info; |
|  | UPDATE   * Update table -  update emp det set city state = ‘salem’ where   city state = ‘trichy’ ; |

3rd week

|  |  |
| --- | --- |
| **GENERAL FUNCTIONS**  [student det table](https://docs.google.com/spreadsheets/d/11Dinr6ocSpjQw_DkyxGYqFO0gHmz8e4P/edit?usp=drivesdk&ouid=101733408306538877494&rtpof=true&sd=true) | -- where clause  select \* from student\_det where city\_state = 'chennai';  -- column filter  select city\_state, student\_name, marks from student\_det where  city\_state = 'chennai';  -- or conditions (same field with different crieteria)  select \* from student\_det where city\_state = 'chennai' or  city\_state = 'madurai' or city\_state = 'salem';  -- and conditions (different fields or multiple fields)  select \* from student\_det where city\_state = 'chennai'  and age >22 ;  -- method 1  select \* from student\_det where (city\_state = 'chennai'  or city\_state ='madurai' or city\_state = 'salem')  and age >20 and marks >50;  -- method2  select \* from student\_det where city\_state in('chennai','madurai',  'salem') and age >20 and marks >50; |
|  | -- mehtod1  select \* from student\_det where (age = 20 or age =21) and marks > 70 ;  -- method 2  select \* from student\_det where age in (20,21,22,23) and marks >70;  select \* from student\_det where age >=20 and marks >70;  select \* from student\_det where city\_state not in('chennai','madurai');  -- <>  select \* from student\_det where city\_state != 'chennai';  -- count  select Count(\*) from student\_det;  select count(student\_name) as Count\_students from student\_det;  -- distinct (create unique fields)  select distinct city\_state from student\_det;  -- unique state count with alias name  select count(distinct city\_state) as State\_unique\_count from student\_det;  -- result\_cat count  select count(distinct Result) as Result\_cat\_count from  student\_det;  -- group by  select city\_state, count(student\_id) as State\_wise\_student\_count from  student\_det group by city\_state;  -- order by  select \* from student\_det order by student\_name desc;  select \* from student\_det order by age desc;  -- limit  select \* from student\_det limit 0,3;  select \* from student\_det limit 4,7;  select \* from student\_det; |
| **TASK** – | -- find out the top 3 position (based on marks)  select\*from student\_det order by marks desc limit 0,3;  -- you need to delete least category students (based on marks)  delete from student\_det order by marks asc limit 1;  -- you can find the last five id's data while using limit function  select\*from student\_det order by student\_id desc limit 0,5;  -- unique Result\_cat count |
| **TASK –** | 1. job unique cat - DISTICNT - select distinct job from emp\_info;  2. Job unique cat count - distinct , count - select count(distinct job) from tanusri.emp\_info;  3. find out the clerk and manager role - select \*from tanu.emp\_info2 where job in ('clerk','manager');  ORDER BY , LIMIT-  1. top 3 pos based on marks - select \* from student\_info order by marks desc limit 0,3;  2. delete least cat students based on marks -  delete from student\_info order by marks asc limit 01;  3. last 5 IDs using limit function - select \* from student\_info order by student\_id desc limit 0,5; |

4th week

|  |  |
| --- | --- |
|  | * Like (\_%) * Not like * Between |
| [Student\_info\_table](https://drive.google.com/file/d/11IEKPkCwjc3SwJs6VNRScYNRRsX8zKBj/view?usp=drivesdk) | SELECT \* FROM tanu.`student\_info for or,and,where function`;  -- like - start with , end , between  -- start with '\_\_%'  SELECT \* FROM tanu.`student\_info for or,and,where function` where student\_name like 've%';  -- end with '%\_\_'  SELECT \* FROM tanu.`student\_info for or,and,where function` where student\_name like '%ni';  -- between '%\_\_%'  select \* from tanu.`student\_info for or,and,where function` where Student\_Name like '%er%';  -- student names marks greater than 50 less than 70  select \* from tanu.`student\_info for or,and,where function` where marks between 50 and 70;  select \* from tanu.`student\_info for or,and,where function` where marks>=50 and marks<= 70;  -- name starting with v , only 5 charact  select \* from tanu.`student\_info for or,and,where function` where student\_name like 'v\_\_\_\_';  -- name starting with a ending with i  select \* from tanu.`student\_info for or,and,where function` where student\_name like 'a%i' ; |
| **TASKS** – | 1. name start with m with 6 charact –   SELECT \* FROM tanu.emp\_info2 where emp\_name like 'm\_\_\_\_\_';   1. name ending with two t's –   select \* from tanu.emp\_info2 where emp\_name like '%tt';   1. name with L in between –   select \* from tanu.emp\_info2 where emp\_name like '%l%'; |
| **TEST** –  [emp info](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\emp_info%20.csv) | 1 -- display table (1)  use tanusri;  SELECT \* FROM tanusri.emp\_info;  2 -- unique job (2)  SELECT distinct job FROM tanusri.emp\_info;  3 -- emp in asc order of sal (3)  SELECT \* FROM tanusri.emp\_info order by salary asc ;  4 -- asc of dept nos and desc of jobs (4)  SELECT \* FROM tanusri.emp\_info order by dept\_no asc , job desc ;  5 -- unique job grp in desc order (5)  SELECT distinct job FROM tanusri.emp\_info order by job desc ;  6 -- emp before 1981 (6)  SELECT \* FROM tanusri.emp\_info where hire\_date<1981;  7 -- emp whose comm>sal (7)  SELECT \* FROM tanusri.emp\_info where Commission>Salary;  8 -- emp asc order of desig who joined in second half of 1981 (8)  select \* from emp\_info;  describe emp\_info;  alter table emp\_info modify hire\_date date;  SELECT \* FROM tanusri.emp\_info where hire\_date between '1981/06/01' and '1981/12/31' order by job asc; |
| [Student\_info\_table](https://drive.google.com/file/d/11IEKPkCwjc3SwJs6VNRScYNRRsX8zKBj/view?usp=drivesdk) | **CALCULATE FUNCTIONS**  1. sum  -- sum  SELECT sum(marks) FROM tanu.`student\_info for or,and,where function`;  2. AVG  SELECT avg(marks) FROM tanu.`student\_info for or,and,where function`;  3. to round of  SELECT round(avg(marks)) FROM tanu.`student\_info for or,and,where function`;  4. for 2 decimals places  SELECT round(avg(marks),2) FROM tanu.`student\_info for or,and,where function`;  5. percentage of marks of individual students ( marks/sum\*100)  note - put comma after star for getting the entire table col  without comma , only the percentage colm will appear  SELECT \*, marks/1865\*100 as marks\_ind FROM tanu.`student\_info for or,and,where function`; |
| **TASKS** –  [emp info](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\emp_info%20.csv) | (SMITH TABLE)   1. sum of sal - select sum(salary) from emp\_info; 2. avg of sal - select avg(salary) from emp\_info; 3. percentage of sal - select \*, salary/29025\*100 as per\_sal FROM tanu.emp\_info2; 4. min - select min(salary) from emp\_info; 5. max of sal - select max(salary) from emp\_info; 6. details of min,max -   -- emp details of minimum salary  select \* from emp\_info order by salary asc limit 0,1;  -- emp details of max salary  select \* from emp\_info order by salary desc limit 0,1; |
| **TOPIC** | * **IF** * **COUNT** * **MULTIPLE IF** * **TRIM** * **GROUP BY** |
|  | May 16  -- if conditions  select \*,if(trim(Att\_status) = 'p','1','0') as present\_update from  att\_info;  --TRIM  use trim if there is space while entering the data  select \*,if(trim(Att\_status) = 'A','1','0') as present\_update from  att\_info;  -- group by  -- working days  select student\_name, count(Att\_date) as  working\_days from att\_info group by student\_name;  -- count present days  select student\_id, student\_name, count(att\_status) as  present\_count from att\_info  where trim(Att\_status) = 'p'group by student\_id, student\_name;  -- count absent days  select student\_id, student\_name, count(att\_status) as  absent\_count from att\_info  where trim(Att\_status) = 'A'group by student\_id, student\_name;  select student\_id, student\_name, count(att\_date) as working\_days,  count(if(trim(att\_status) = 'p','1',null)) present,  count(if(trim(att\_status) = 'A','1',null))  as absent\_count from att\_info group by student\_id, student\_name;  -- Multiple if  create table student\_marks\_results  as select \*,if(average>=35,"Pass","Fail") as Results,  if(Tamil>=35 and english >=35 and  accounts >=35 and commerce >=35 and economics  >=35,"pass", "Fail") as Ind\_results  from student\_marks1; |
| **TASK-**  [**logical functions**](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\logical%20func.csv) | 1. result on basis of avg - above 35% = pass  select \*, if(average>35,'p','f') as result from tanusri.`logical func`;  2. individual results - of each subject marks , above 35 = pass  select \*, if(tamil>35 and english>35 and accounts>35 and commerce>35 and economics >35 , 'p','f') as aggregate  from tanusri.`logical func`;  3. delete total avg and cal avg again  - alter table tanusri.`logical func` drop column total;  - alter table tanusri.`logical func` drop column average;  - SELECT \* FROM tanusri.`logical func`;  - select \*, (tamil+english+accounts+commerce+economics) as total from tanusri.`logical func`;  - select \*, (tamil+english+accounts+commerce+economics)/5 as average from tanusri.`logical func`;  **NOTE** -- for sum,avg,  1. to calculate horizontally = (col1+col2)  (col1+col2)/2  2. to calculate vertically = sum(col1 and col2)  avg(col1 and col2)  Here , the total and avg are self join col . To join it into the table and create a new one  create table student\_mark as select \*, (tamil+english+accounts+commerce+economics) as total,  (tamil+english+accounts+commerce+economics)/5 as average  from tanusri.`logical func`; |

5th week

|  |  |
| --- | --- |
|  | select \* from student\_info;  --Count  select city\_state, count(student\_id) as student\_count  from student\_info group by city\_state;  select city\_state, sum(marks) as Total\_marks  from student\_info group by city\_state;  select result,count(student\_id)from student\_info group by  result;  select age,count(student\_id) as student\_count  from student\_info group by age order by student\_count  desc ;  create table state\_wise\_stu\_report as select city\_state,  count(student\_id) as student\_count,  sum(marks) as Total\_marks, round(avg(marks),0) as Avg\_marks  from student\_info  group by city\_state ;  show tables;  select \* from state\_wise\_stu\_report where avg\_marks>80; |
| **TEST** | 1. List the employees with their annual salary and daily salary > 100 2. List the employees who are either clerk or analyst on desc order 3. List the employees who joined on 1-may-81 , 3-dec-81 , 17-dec-81 , 19-jan-80 in asc order of seniority 4. List the emp who are working for dept no 10 or 20 5. List the employees who joined in the year 81 6. List the employees whose annual salary ranging from 22000 and 45000 7. A. create a field income criteria – if income >15000 – good income   if income <15000 – bad income  B. create a field income criteria2 – if income >15000 and exp < 10000 – good income  Else - bad income  C. . create a field income criteria3 – if income >20000 or exp < 15000 -- avg income else bad income |
|  | **STRING FUNCTIONS –**   * LCase * UCase * Left * Right * Concat * Trim * Char\_Length * Mid * Length |
| create table student\_dett (student\_id int, student\_name varchar(35),  student\_initial varchar(40), marks int, gender varchar(45),  primary key(Student\_id));  insert into student\_dett values  (14006, 'Marc Crier', 'S', 71, 'Female'),  (14007, 'William Brown', 'D', 42, 'Male'),  (14008, 'Matt Hagelstein', 'P', 77, 'Female'),  (14009, 'Toby Gnade', 'V', 89, 'Male'),  (14010, 'Claire Gute', 'D', 79, 'Female'),  (14011, 'Darrin Van Huff', 'P', 75, 'Female'); | select \* from student\_dett;  -- lcase  select student\_id,lcase(student\_name) as student\_name,  student\_initial, marks , gender from student\_dett;  -- ucase  select student\_id,ucase(student\_name) as student\_name,  student\_initial, marks , gender from student\_dett;  -- left  select left(student\_id,2) as def\_id, right(student\_id,3) as student\_id from student\_dett;  -- concat (merge the fields)  select concat(student\_name,".",student\_initial) as student\_full\_name from student\_dett;  -- 4 letter second name  select \*,right(student\_name,4) as student\_first\_name from student\_dett where student\_name like '% \_\_\_\_'; |
| **TASK** | -- names that has vowels  select \* from student\_dett where student\_name like '%a%' or student\_name like '%e%' or student\_name like '%i%' or student\_name like '%o%' or student\_name like '%u%'; |
|  | -- character length  select \* from student\_det where char\_length(student\_name)=4;    -- find the name starting with s and  end with r charact being 8  select \* from student\_det where student\_name like 's%r' and length(student\_name)=8 ;  -- mid mid(column name , where to start ,how many elements to select)  select mid(student\_name,2,2) as mid\_name from student\_det ;  select mid(student\_id,3,2) as mid\_id from student\_det ; |
|  | **Date Functions**   * Date ADD * Datediff * Timestamp DIff * Date Format * Year * Day * Month * Now |
| [emp info](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\emp_info%20.csv) | -- year  SELECT \* FROM besant10.emp\_info2 where year(hire\_date) = '1981';  -- month  SELECT \* FROM besant10.emp\_info2 where year(hire\_date) ='1981' and month(hire\_date) = '12' ;  -- day  SELECT \* FROM besant10.emp\_info2 where day(hire\_date) ='28' and month(hire\_date) = '09' ;  -- ALTERNATE METHOD USING LIKE  SELECT \* FROM besant10.emp\_info2 where hire\_date like '1981-12%';  SELECT \* FROM besant10.emp\_info2 where hire\_date like '\_\_\_\_-09-\_\_';  SELECT \* FROM besant10.emp\_info2 where hire\_date like '\_\_81-09-28'; |
| **TASK** | year - 2 char like  month - name %b  day - 11th %D  create table emp\_det6 as select \*, date\_format(hire\_date , '%y') as year\_name , date\_format(hire\_date , '%b') as month\_name ,  date\_format(hire\_date , '%D') as date\_name from  besant10.emp\_info2 ;  select \* from emp\_det6;  select concat(year\_name,"-",month\_name,"-",date\_name) from emp\_det6;  select\*, date\_format( hire\_date , '%y-' '%b-' '%D' ) as HIRE\_DATE from besant10.emp\_info2 ; |
| [emp det](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\Emp_det.csv) | -- now - gives the present time and date  select \*, now() from emp\_det;    -- curdate - only the present date  select \*, curdate() from emp\_det;  -- timestampdiff - diff between the years . ex. years of exp  a. in years  select \*, timestampdiff(year,hire\_date , curdate()) as experience from emp\_det;  b. in days  select \*, timestampdiff(day,hire\_date , curdate()) as experience from emp\_det;  c. in months  select \*, timestampdiff(month,hire\_date , curdate()) as experience from emp\_det;  To add years , months , days beside it  a. for days  select \*, concat(timestampdiff(day,hire\_date , curdate())," days") as experience from emp\_det;    b. for months  select \*, concat(timestampdiff(month,hire\_date , curdate())," months") as experience from emp\_det;  NOTE - to get a space or - , "-"  -- date add - to find the no. of months,years,days from the current date  A. 22 months from hire date  select\*, date\_add( hire\_date , interval 22 month) as year\_add from emp\_det;  B. 22 months from current date  select\*, date\_add( curdate() , interval 22 month) as year\_add from emp\_det; |

6th Week

|  |  |
| --- | --- |
|  | **JOINS**  - Inner Join - matching similar rows  - Left Join - add the infos to left table , null in right table  - Right Join - add the infos to right table , null in left table  - Cross join  - Full Outer Join - union of left and right table |
|  | -- inner join (lefttable.commonfield = righttable.commonfield)  select \* from emp\_det1 inner join sal\_info on emp\_det1.emp\_id = sal\_info.emp\_id;  select \* from sal\_info inner join emp\_det1 on sal\_info.emp\_id = emp\_det1.emp\_id ;  -- left join  select \* from emp\_det1 left join sal\_info on emp\_det1.emp\_id = sal\_info.emp\_id;  -- right join  select \* from emp\_det1 right join sal\_info on emp\_det1.emp\_id = sal\_info.emp\_id ;  insert into sal\_info values  (181,8,'2022-07-23' ,21000);  -- union ( left join query union right join query )  select \* from emp\_det1 left join sal\_info on emp\_det1.emp\_id = sal\_info.emp\_id union  select \* from emp\_det1 right join sal\_info on emp\_det1.emp\_id = sal\_info.emp\_id ;  -- No repetition of a column  select emp\_det1.emp\_id, emp\_det1.emp\_name , emp\_det1.designation , emp\_det1.date\_of\_join , sal\_info.sal\_id , sal\_info.sal\_date , sal\_info.amount  from emp\_det1 inner join sal\_info on emp\_det1.emp\_id = sal\_info.emp\_id ; |
| **TASK**  [student data table](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\Student_data.csv)  [course ID table](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\Course_ID.csv)  [marks det table](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\Marks_det.csv) | 1. 3 table connection (select \* from Table1 inner join Table 2 on Table1.similarfieldin T1&T2 = Table2.similarfieldin T1&T2 inner join  Table3 on Table2.similarfieldin T2&T3 = Table3.similarfieldin T2&T3; )  select \* from course\_id inner join student\_data on course\_id.course\_id = student\_data.course\_id inner join  marks\_det on student\_data.Student\_ID  = marks\_det.student\_id;  2. 1st table ID , NAME , STATE , AGE , COMMUNITY  2nd table ID,NAME  3rd table MARK\_ID , FINANCE , COST CORP  select course\_id.course\_id , course\_id.Course\_Name , student\_data.Student\_ID , student\_data.Student\_Name , student\_data.City\_State  , student\_data.age , student\_data.Community , marks\_det.Marks\_ID , marks\_det.Finance , marks\_det.Cost , marks\_det.Corporate from  course\_id inner join student\_data on course\_id.course\_id = student\_data.course\_id inner join  marks\_det on student\_data.Student\_ID= marks\_det.student\_id; |
| **TEST**  [profit source table](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\PROFITSOURCE.csv)  [products table](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\PRODUCTS.xlsx) | 1. find the mismatch btw the tables ( union join )  select profitsource.PVT\_Limited\_Companies , profitsource.Order\_No , profitsource.City , profitsource.Income , profitsource.Expenses,  profitsource.Total\_Profit , products.Category , products.Product\_Name , products.Date\_Of\_Purchase  profitsource from profitsource left join products on profitsource.Order\_No = products.Order\_No union  select profitsource.PVT\_Limited\_Companies , profitsource.Order\_No , profitsource.City , profitsource.Income , profitsource.Expenses,  profitsource.Total\_Profit , products.Category , products.Product\_Name , products.Date\_Of\_Purchase from profitsource right join products on profitsource.Order\_No = products.Order\_No ;    2. join the missing fields(inc,exp, profit) from surce table to Vlook up table  select besant10.vlookup.order\_no , besant10.vlookup.PVT\_Limited\_Companies ,  besant10.vlookup.city , besant10.vlookup.category ,  besant10.vlookup.product\_name , besant10.vlookup.date\_of\_purchase , besant10.source.income , besant10.source.Expenses , besant10.source.Total\_Profit  from besant10.vlookup inner join besant10.source on besant10.vlookup.PVT\_Limited\_Companies = besant10.source.PVT\_Limited\_Companies;   1. D/B full outer join & inner join 2. usage of Group by function and list out its types 3. D/B btw timestamp& Timestamp diff |
| **TASK** [employees table](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\employees.csv)  [departments table](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\departments.csv) | which depts do not have managers with no duplicates |
|  | select course\_info.course\_id, course\_info.course\_name,mark\_info.marks\_id, mark\_info.finance,mark\_info.cost,mark\_info.corporate,student\_data.student\_id,student\_data.student\_name,  student\_data.age,student\_data.city\_state,student\_data.community  from mark\_info inner join student\_data on mark\_info.student\_id = student\_data.Student\_ID inner join course\_info on student\_data.Course\_ID = course\_info.course\_id;  -- total marks  select course\_info.course\_id, course\_info.course\_name,mark\_info.marks\_id, mark\_info.finance, mark\_info.cost,  mark\_info.corporate,(finance+cost+corporate) as total\_marks,student\_data.student\_id,student\_data.student\_name,  student\_data.age,student\_data.city\_state,student\_data.community  from mark\_info inner join student\_data on mark\_info.student\_id = student\_data.Student\_ID inner join course\_info on student\_data.Course\_ID = course\_info.course\_id;    -- average, round off, results using case when, individual results  select course\_info.course\_id, course\_info.course\_name,mark\_info.marks\_id, mark\_info.finance, mark\_info.cost,  mark\_info.corporate,(finance+cost+corporate) as total\_marks,round((finance+cost+corporate)/3,0)as average,  case when round((finance+cost+corporate)/3,0)>=35 then "pass"else "fail" end as results,case when finance>= 35 and cost>= 35 and corporate>=35  then "pass" else "fail" end as individual\_results,  student\_data.student\_id,student\_data.student\_name,  student\_data.age,student\_data.city\_state,student\_data.community  from mark\_info inner join student\_data on mark\_info.student\_id = student\_data.Student\_ID inner join course\_info on student\_data.Course\_ID = course\_info.course\_id; |
|  | -- Create grade field with these criteria  Finance<35 or Cost<35 or Corporate<35 - No Grade  avg>90 - A Grade  avg>80 - B Grade  avg>70 - C Grade  35>avg<70 - D Grade  select \* , case  when Finance < 35 or Cost<35 or Corporate <35 then "no\_grade"  when average>70 then "C\_grade"  when average>80 then "B\_grade"  when average>90 then "A\_grade" else "D\_grade" end as grade from stu\_det1;  -- OR  select \* , case  when Finance >= 35 and Cost>=35 and Corporate >=35 then  case  when average>70 then "C\_grade"  when average>80 then "B\_grade"  when average>90 then "A\_grade" else "D\_grade" end else "no\_grade" end as grade from stu\_det1 ;  -- COURSE WISE PASS REPORT  - when the IND\_RESULT field is self join, use HAVING  select new.course\_id.course\_id , new.course\_id.Course\_Name ,  new.student\_data.Student\_ID , new.student\_data.Student\_Name , new.student\_data.City\_State  , new.student\_data.age , new.student\_data.Community , new.marks\_det.Marks\_ID ,  new.marks\_det.Finance ,new.marks\_det.Cost , new.marks\_det.Corporate , (finance+cost+corporate) as total\_marks ,  round( (finance+cost+corporate)/3) as average , case when round( (finance+cost+corporate)/3) >=35 then "pass" else "fail" end as result ,  case when Finance >=35 and Cost >=35 and Corporate >=35 then "pass" else "fail"  end as ind\_result from  new.course\_id inner join new.student\_data on new.course\_id.course\_id = new.student\_data.course\_id inner join  new.marks\_det on new.student\_data.Student\_ID= new.marks\_det.student\_id where new.course\_id.course\_name = 'CS' having ind\_result= 'pass';    select new.course\_id.course\_id , new.course\_id.Course\_Name ,  new.student\_data.Student\_ID , new.student\_data.Student\_Name , new.student\_data.City\_State  , new.student\_data.age , new.student\_data.Community , new.marks\_det.Marks\_ID ,  new.marks\_det.Finance ,new.marks\_det.Cost , new.marks\_det.Corporate , (finance+cost+corporate) as total\_marks ,  round( (finance+cost+corporate)/3) as average , case when round( (finance+cost+corporate)/3) >=35 then "pass" else "fail" end as result ,  case when Finance >=35 and Cost >=35 and Corporate >=35 then "pass" else "fail"  end as ind\_result from  new.course\_id inner join new.student\_data on new.course\_id.course\_id = new.student\_data.course\_id inner join  new.marks\_det on new.student\_data.Student\_ID= new.marks\_det.student\_id where new.course\_id.course\_name = 'CA' having ind\_result= 'pass';  - use AND if the ind\_result field is part of the table    select \* , case  when Finance < 35 or Cost<35 or Corporate <35 then "no\_grade"  when average>70 then "C\_grade"  when average>80 then "B\_grade"  when average>90 then "A\_grade" else "D\_grade" end as grade from stu\_det1 where course\_name = 'CS' and ind\_result= 'pass'; |
| **TEST**  [emp det](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\Emp_det.csv)  [dept det](file:///C:\Users\TANU%20FAB\OneDrive\Desktop\SQL\Dep_det.csv) | 1. Join emp det table and dep det table without duplicates and show dep\_name = research and accounts in ascending order 2. Join the tables and create a field experience ; job = manager , analyst ; location = newyork , dallas ; experience > 40 ; community = null ; ascending order of location |

7th week

|  |  |
| --- | --- |
|  | **Triggers**  - is a stored program which is automatically executed or fired when some event occurs  TYPES-  1. row level - triggers an event at each row updated , inserted or deleted  2. statement level - triggers an event at at table level for each sql statement  - creates a new table by applying the triggers |
|  | create table student\_det (student\_id int , student\_name varchar(30) , city\_state varchar(30),age int ,  community varchar(30), marks int , primary key ( student\_id));  delimiter -  create trigger age\_verify before insert on student\_det for each row  begin  if new.age<=0 then set new.age=0 ; end if;  end -  delimiter ;  -- in the above query , - is the delimiter  insert into student\_det values  (1 ,'Vasanth', 'Erode', 21, 'BC', 32),  (2 ,'Guru', 'Tiruppur', 20 ,'MBC' ,78),  (3 ,'Gokul', 'Tiruchirapalli', -18, 'BC', 89),  (4 ,'Mani', 'Kumarapalayam', -24, 'BC', 56);  select \* from student\_det;  -- community = null , give update\_your\_community  delimiter /  create trigger community\_verify before insert on student\_det for each row  begin  if new.community='null' then set new.community='update\_your\_community' ; end if;  end  / delimiter ;  insert into student\_det values  (5, 'Moorthy', 'Salem', 18, 'MBC', 90),  (6, 'Amutha', 'Chennai', 17, 'null', 43),  (7, 'Jaga' ,'Madurai', 24, 'BC', 23);  -- or  delimiter /  create trigger community\_verify before insert on student\_det for each row  begin  if new.community is null then set new.community='update\_your\_community' ; end if;  end  / delimiter ;  insert into student\_det values  (5, 'Moorthy', 'Salem', 18, 'MBC', 90),  (6, 'Amutha', 'Chennai', 17, null, 43),  (7, 'Jaga' ,'Madurai', 24, 'BC', 23);  select \* from student\_det;  -- marks<35 then give reappear  alter table student\_det modify marks varchar(30) ;  delimiter --  create trigger mark\_verify before insert on student\_det for each row  begin  if new.marks<=35 then set new.marks='reappear' ; end if ;  end -- delimiter ;  insert into student\_det values  (8, 'Murthy', 'Salem', 18, 'MBC', 90),  (9, 'Anu', 'Chennai', 17, 'null', 43),  (10, 'Jagan' ,'Madurai', 24, 'BC', 23);  select \* from student\_det; |
|  | create table emp\_det ( emp\_id int , emp\_name varchar(30) , hire\_date date , mail\_id int ) ;  create table comment\_det ( comment\_id int , remarks varchar(60) ) ;  delimiter //  create trigger mail\_id after insert on emp\_det for each row  begin  if new.mail\_id is null then insert into comment\_det(comment\_id,remarks) values (new.emp\_id,concat(' hi ' , new.emp\_name ,  ' kindly update\_your mail\_id')) ; end if ;  end //  delimiter ;  -- multipe if  create table emp\_det2 ( emp\_id int , emp\_name varchar(30) , daily\_production int ) ;  create table salary\_det ( emp\_id int , salary int ) ;    delimiter //  create trigger salary\_verify after insert on emp\_det2 for each row  begin  if  new.daily\_production>100 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,1000) ;  elseif new.daily\_production>80 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,750);  elseif new.daily\_production>65 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,500);  elseif new.daily\_production>40 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,400);  elseif new.daily\_production>0 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,250); end if ; end //  delimiter ;  -- using case when  delimiter ==  create trigger salary\_verify after insert on emp\_det2 for each row  case  when new.daily\_production>100 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,1000) ;  when new.daily\_production>80 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,750) ;  when new.daily\_production>65 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,500) ;  when new.daily\_production>40 then insert into salary\_det(emp\_id,salary) values (new.emp\_id,400) ;  else insert into salary\_det(emp\_id,salary) values (new.emp\_id,250);  end case ;  end ==  delimiter ; |
|  | **STORED PROCEDURES**  -- procedure creation  delimiter --  create procedure emp\_data()  begin  select \* from emp\_det where job in ( 'MANAGER', 'ANALYST');  select \* from emp\_det where salary > 2000 ;  select dept\_no, count(emp\_id) as emp\_count from emp\_det group by dept\_no;  end --  delimiter ;  call emp\_data(); |
|  | **Different Types of Database Keys**   * Candidate Key * Primary Key * Super Key * Alternate Key * Foreign Key   create table dep\_info ( dept\_no int , dep\_name varchar(50) , location varchar(55) , primary key(dept\_no));  create table emp\_info ( emp\_id int , emp\_name varchar(50) ,job varchar(55) , mgr\_id int ,  hire\_date date , salary int , commission int ,dept\_no int , primary key(dept\_no) ,  constraint dep\_for\_key foreign key (dept\_no) references dep\_info(dept\_no)) ;  DROP TABLE DEP\_INFO;   * Composite Key |
|  | **TRANSACTIONS CONTROL LANGUAGE** |
|  | Commit - save point  Rollback - undo changes |