**Session 2 : Keys Lab Session:**

-- Keys Lab Session:

-- Create a database School:

create database school;

-- Create some tables for school database

use school;

create table batches(

batch\_id int PRIMARY KEY,

batch\_name varchar(50) NOT NULL);

desc batches;

-- Inserting some dummy data in batches table:

insert into batches(batch\_id, batch\_name) values

(1, 'A'),

(2, 'B'),

(3, 'C'),

(4, 'D');

-- Error Code: 1062. Duplicate entry '1' for key 'batches.PRIMARY'

-- Create students table:

-- student\_id, first\_name, last\_name

create table students(

student\_id int auto\_increment PRIMARY KEY,

first\_name varchar(50) not null,

last\_name varchar(50) not null,

batch\_id int,

FOREIGN KEY(batch\_id) references batches(batch\_id) ON DELETE CASCADE ON UPDATE CASCADE);

-- Error Code: 1060. Duplicate column name 'first\_name'

-- Inserting some dummy data in students:

insert into students(first\_name, last\_name, batch\_id) values

('John', 'Cena', 1),

('Will', 'Smith', 2),

('Jim', 'Brown', 1),

('Jack', 'Johnson', 3),

('Christiano', 'Messi', 1);

-- Some eureka moments:

update batches

set batch\_id = 2

where batch\_id = 4;

-- Error Code: 1062. Duplicate entry '2' for key 'batches.PRIMARY'

update students

set batch\_id = 3

where batch\_id = 2;

-- Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`school`.`students`, CONSTRAINT `students\_ibfk\_1` FOREIGN KEY (`batch\_id`) REFERENCES `batches` (`batch\_id`))

delete from batches

where batch\_id = 2;

-- Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`school`.`students`, CONSTRAINT `students\_ibfk\_1` FOREIGN KEY (`batch\_id`) REFERENCES `batches` (`batch\_id`))

delete from batches

where batch\_id = 4;

DELETE FROM BATCHES

WHERE batch\_id = 2;

-- HW: Read about alter command.

-- HW: Read about SQL datatypes.

delete from batches;

-- Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.

-- Select command:

select student\_id

from students;

select student\_id, first\_name, batch\_id

from students;

select \*

from students;

**Session 3: CRUD:**

**-- CRUD Queries:**

**use sakila;**

**desc film;**

**INSERT INTO film**

**VALUES (default, 'Comedy movie', 'hehehe', 2022, 1, NULL, 3, 3.39, 152, 19.99, 'G', 'Trailers', default);**

**-- Select query:**

**select 1;**

**select 'Hello world';**

**select 'Rahul' as output;**

**select 'Rahul' output;**

**select \***

**from film;**

**select film\_id, title**

**from film;**

**use school;**

**select 1**

**from students;**

**use sakila;**

**select rental\_duration**

**from film;**

**-- Print unique rental\_duration in films table:**

**select distinct rental\_duration**

**from film;**

**-- Question: Get all distinct raings per year in films data:**

**select release\_year, rating**

**from film;**

**select distinct release\_year, rating**

**from film;**

**select release\_year, distinct rating**

**from film;**

**-- Problem satement: Get all the movies with PG-13 rating.**

**select title, release\_year, rating**

**from film**

**where rating = 'Pg-13';**

**-- Logical Operators:**

**-- Problem statement: Get all the films which were released in 2006 and have rating Pg-13?**

**select title, release\_year, rating**

**from film**

**where release\_year = 2006 and rating = 'PG-13';**

**-- Problem statement: Get all the films which were either released in 2006 or have rating Pg-13?**

**select title, release\_year, rating**

**from film**

**where release\_year = 2006 or rating = 'PG-13';**

**-- Problem statement: Get all the films which were released in 2006 and have other than rating Pg-13?**

**select title, release\_year, rating**

**from film**

**where release\_year = 2006 and rating != 'PG-13';**

**select title, release\_year, rating**

**from film**

**where release\_year = 2006 and rating <> 'PG-13';**

**select title, release\_year, rating**

**from film**

**where release\_year = 2006 and not rating = 'PG-13';**

**-- Order by:**

**-- Problem statement: Get all the movies sorted according to rental\_duration:**

**select film\_id, title, rental\_duration**

**from film**

**order by rental\_duration;**

**select film\_id, title, rental\_duration, rental\_rate**

**from film**

**order by rental\_duration, rental\_rate desc;**

**-- In clause:**

**-- Problem statement: Get all the students which belongs to either of batches: 1, 4, 5, 3, 6, 7, 8**

**use school;**

**select \***

**from students**

**where batch\_id in(1, 4, 5, 3, 6, 7, 8);**

**-- Between Operator:**

**-- Problem statement: Get data of all the films having rental\_duration between 1 to 5:**

**use sakila;**

**select title, rental\_duration**

**from film**

**where rental\_duration between 3 and 5**

**order by rental\_duration;**

**-- Alter Query:**

**use school;**

**alter table students**

**add psp int;**

**-- Update Query?**

**update students**

**set last\_name = 'Ronaldo'**

**where student\_id = 5;**

**set sql\_safe\_updates = 0;**

**set sql\_safe\_updates = 1;**

**show variables like 'sql\_safe\_updates';**

**-- Delete vs Truncate vs Drop:**

**-- Delete:**

**delete from students**

**where student\_id = 5;**

**-- Truncate:**

**truncate table students;**

**-- Drop:**

**drop table students;**

**use sakila;**

**select distinct special\_features, rental\_duration, rental\_rate**

**from film**

**order by special\_features;**

**desc film;**

**Session 4 : CRUD Lab Session:**

**Extra Questions:** [**Extra Questions**](https://docs.google.com/document/d/1rkUWrwZzT0tbtKv64YSJAAjStwgre9Sx2OWarbvbYsU/edit?tab=t.0)

**use sakila;**

**select \***

**from film**

**limit 2**

**offset 99;**

**-- Like keywords:**

**-- Get all the movies which have love in it?**

**select title**

**from film**

**where binary title like '%LOVE%';**

**-- Working with Null values:**

**select 1 = 1;**

**select 1 = 2;**

**select 1 = Null;**

**select Null = Null;**

**select Null is Null;**

**-- Problem 1: Identifying Popular Films**

**-- The management wants to identify the first 10 distinct film titles from the**

**-- film table that contain the word "adventure" in their title (case insensitive),**

**-- sorted alphabetically. Write an SQL query that returns the film titles.**

**select distinct title**

**from film**

**where title like '%adventure%'**

**order by title**

**limit 10;**

**-- Problem 2: Tracking Unreturned Rentals**

**-- The store is experiencing issues with customers not returning their rentals on time.**

**-- From the rental table, list the first 5 rental IDs where the return\_date is NULL,**

**-- ordered by the rental\_id in descending order. Write an SQL query to fetch this data.**

**select rental\_id**

**from rental**

**where return\_date is null**

**order by rental\_id desc**

**limit 5;**

**select \***

**from rental**

**where rental\_id = 11739;**

**use school;**

**select concat(first\_name, ' ', last\_name)**

**from students;**

**select round(12.3343444, 2);**

**Session 5: Joins:**

**-- Joins:**

**-- Get name of students and the name of batches which are assigned to them?**

**use school;**

**select students.first\_name, students.last\_name, batches.batch\_name**

**from students**

**join batches**

**on students.batch\_id = batches.batch\_id;**

**select s.first\_name, s.last\_name, b.batch\_name**

**from students as s**

**join batches as b**

**on s.batch\_id = b.batch\_id;**

**select s.first\_name, s.last\_name, b.batch\_name**

**from students s**

**join batches b**

**on s.batch\_id = b.batch\_id;**

**-- Joining multiple tables:**

**-- For every actor get the name of mvoies in which they acted.**

**-- Find out the tables from which we need our data.**

**-- film, actor, film\_actor**

**-- Order of joining of these tables?**

**-- film -> film\_actor -> actor**

**use sakila;**

**select concat(a.first\_name, ' ', a.last\_name) full\_name, f.title**

**from film f**

**join film\_actor fa**

**on f.film\_id = fa.film\_id**

**join actor a**

**on a.actor\_id = fa.actor\_id;**

**-- Left Join:**

**-- Get all the students along with their batch names. Get students with unassigned**

**-- batches as well.**

**-- 2nd part: Find students for whom batches aren't assigned?**

**select \***

**from students s**

**left join batches b**

**on s.batch\_id = b.batch\_id**

**where b.batch\_id is null;**

**-- Right Join:**

**-- Get all the batch along with the students being assigned to them. Also get those batches**

**-- for whom no studetns are assigned?**

**select \***

**from students s**

**right join batches b**

**on s.batch\_id = b.batch\_id;**

**-- Get all the batches for whom no students are assigned?**

**select \***

**from students s**

**right join batches b**

**on s.batch\_id = b.batch\_id**

**where s.student\_id is null;**

**select b.\***

**from students s**

**right join batches b**

**on s.batch\_id = b.batch\_id**

**where s.student\_id is null;**

**select b.\***

**from batches b**

**left join students s**

**on s.batch\_id = b.batch\_id**

**where s.student\_id is null;**

**-- Aggregate Functions:**

**select count(batch\_id)**

**from students;**

**-- Find total number of students?**

**select count(student\_id)**

**from students;**

**select count(\*)**

**from students;**

**select count(batch\_id)**

**from students;**

**-- What is max psp in students table:**

**select max(psp)**

**from students;**

**select avg(psp)**

**from students;**

**-- aggregate(aggregate)**

**-- select max(avg(psp))**

**-- from students;**

**select min(psp), sum(psp)**

**from students;**

**select avg(batch\_id)**

**from students;**

**select sum(batch\_id)/count(\*)**

**from students;**

**-- Problem Statement: Display a list of customers who rented a film in 'Horror' category**

**-- Include name, last\_name, email and film they rented.**

**-- Step 1: Get all tables that you need to join.**

**-- Step 2: Get the order in which we should join them?**

**select \***

**from students**

**cross join batches;**

**Session 7: Subquery:**

**-- Subqueries:**

**-- Group by and Having:**

**use school;**

**-- Problem statement: Get avg(psp) of every learners:**

**select avg(psp)**

**from students;**

**-- 83.333**

**select avg(psp), batch\_id**

**from students**

**group by batch\_id;**

**select avg(psp), batch\_id**

**from students**

**group by batch\_id;**

**-- Find all the batches which have avg(psp) > 80.**

**select avg(psp), batch\_id**

**from students**

**group by batch\_id**

**having avg(psp) > 80;**

**-- HW: Find all the batches which have avg(psp) > 80. Consider those students having**

**-- psp > 70 only.**

**-- Problem statement: Find all the students having psp > psp of s\_id = 2?**

**-- Step 1: Find psp of s\_id = 2.**

**select psp**

**from students**

**where student\_id = 2;**

**-- 75 -> x**

**-- Step 2: Find all the students having psp > x.**

**select \***

**from students**

**where psp > (**

**select psp**

**from students**

**where student\_id = 2);**

**-- Problem statement: Find data of students having psp > min(psp) of every batch\_id(2)**

**-- Step 1: Find min(psp) of b\_id = 2:**

**select min(psp)**

**from students**

**where batch\_id = 2;**

**-- 75 -> x**

**-- Step 2: Find all the students having psp > x**

**select \***

**from students**

**where psp > (**

**select min(psp)**

**from students**

**where batch\_id = 2);**

**-- Find all the years where avg(rental\_rate) > Global avg(rental\_rate).**

**use sakila;**

**-- Step 1: Finding global avg(rental\_rate)**

**select avg(rental\_rate)**

**from film;**

**-- 2.9818 -> x**

**-- Step 2: Find all years with avg(rental\_rate) > x.**

**select avg(rental\_rate), release\_year**

**from film**

**group by release\_year;**

**-- 2006 -> 2.980**

**-- 2022 -> 3.39**

**-- 2025 -> 4.39**

**select release\_year**

**from film**

**group by release\_year**

**having avg(rental\_rate) > (**

**select avg(rental\_rate)**

**from film);**

**-- Subquery inside from clause:**

**-- Find all the students having psp > min(psp) among avg(psp) of every batch?**

**use school;**

**select avg(psp)**

**from students;**

**-- Step 1: Find avg(psp) of every batch. -> x**

**select avg(psp) as psp**

**from students**

**group by batch\_id;**

**-- Step 2: Find min(x)**

**select min(psp)**

**from (**

**select avg(psp) as psp**

**from students**

**group by batch\_id) as xyz;**

**-- 72 -> y**

**-- Step 3: Find all students having psp > y**

**select \***

**from students**

**where psp > (**

**select min(psp)**

**from (**

**select avg(psp) as psp**

**from students**

**group by batch\_id) as xyz);**

**-- All and Any:**

**-- Find all the learners where psp > min(psp) of all batches?**

**-- Step 1: Find min(psp) of every batch.**

**select min(psp)**

**from students**

**group by batch\_id;**

**-- (72, 73, 75, 92) -> x**

**-- Step 2: Find all the students where psp > x**

**select \***

**from students**

**where psp >= All(**

**select min(psp)**

**from students**

**group by batch\_id);**

**-- Find all the learners where psp > min(psp) of any batches?**

**select \***

**from students**

**where psp >= Any(**

**select min(psp)**

**from students**

**group by batch\_id);**

**-- Co-related Subqueries:**

**-- Find all the students where psp > avg(psp) of their batch.**

**-- Step 1:**

**select avg(psp)**

**from students**

**where batch\_id = 1;**

**-- 81**

**select avg(psp)**

**from students**

**where batch\_id = 2;**

**-- 75**

**-- Step 2:**

**select \***

**from students s**

**where psp > (**

**select avg(psp)**

**from students**

**where batch\_id = s.batch\_id);**

**-- Exists:**

**use sakila;**

**-- Find actors who have acted in atleast one movie:**

**select \***

**from film\_actor**

**where actor\_id = 10;**

**select \***

**from actor a**

**where exists(**

**select \***

**from film\_actor**

**where actor\_id = a.actor\_id);**

**Session 8: Indexing:**

**-- Indexing:**

**-- Syntax of creating an index:**

**-- create index idx\_col**

**-- on table\_name(col);**

**-- Syntax for dropping an index:**

**-- drop index index\_name**

**-- on table\_name;**

**use sakila;**

**desc actor;**

**-- actor\_id, last\_name**

**select \***

**from actor**

**where first\_name = 'Rahul';**

**-- Query execution plan:**

**-- Query execution plan without indexed column. -> first\_name**

**explain select \***

**from actor**

**where first\_name = 'Rahul';**

**-- Query execution plan with indexed column -> last\_name**

**explain select \***

**from actor**

**where last\_name = 'Janghu';**

**-- Query execution time difference on search based on indexed vs non indexed column:**

**-- Without indexing:**

**explain analyze select \***

**from actor**

**where first\_name = 'Rahul';**

**-- Table scan on actor (cost=20.2 rows=200) (actual time=0.55..0.622 rows=200 loops=1)**

**-- With indexed column:**

**explain analyze select \***

**from actor**

**where last\_name = 'Janghu';**

**-- Index lookup on actor using idx\_actor\_last\_name (last\_name='Janghu') (cost=0.35 rows=1) (actual time=0.103..0.125 rows=1 loops=1)**

**-- Multiple indexes used in search:**

**explain select \***

**from actor**

**where first\_name = 'Rahul' and last\_name = 'Janghu';**

**desc actor;**

**-- actor\_id, last\_name**

**explain select \***

**from actor**

**where actor\_id = 15 and last\_name = 'Janghu';**

**-- Indexing on strings:**

**explain analyze select \***

**from film**

**where title = 'ADVENTURE TRIP';**

**-- Index lookup on film using idx\_title (title='ADVENTURE TRIP') (cost=0.35 rows=1) (actual time=0.44..0.447 rows=1 loops=1)**

**-- Analyze execution time without indexing:**

**drop index idx\_title**

**on film;**

**explain analyze select \***

**from film**

**where title = 'ADVENTURE TRIP';**

**-- Table scan on film (cost=103 rows=1000) (actual time=0.186..1.45 rows=1002 loops=1)**

**-- Indxing on 1st character:**

**create index idx\_title**

**on film(title(1));**

**explain analyze select \***

**from film**

**where title = 'ADVENTURE TRIP';**

**-- Index lookup on film using idx\_title (title='ADVENTURE TRIP') (cost=13.6 rows=46) (actual time=0.13..0.349 rows=46 loops=1)**

**-- Index on first 2 characters:**

**drop index idx\_title**

**on film;**

**create index idx\_title**

**on film(title(2));**

**explain analyze select \***

**from film**

**where title = 'ADVENTURE TRIP';**

**-- Index lookup on film using idx\_title (title='ADVENTURE TRIP') (cost=0.7 rows=2) (actual time=0.0433..0.0507 rows=2 loops=1)**

**-- Repeat this same process and take readings for indexing on:**

**-- first 3 characters**

**-- first 4 characters**

**-- first 5 characters**

**-- Using:**

**use school;**

**select s.first\_name, b.batch\_name**

**from students s**

**join batches b**

**on s.batch\_id = b.batch\_id;**

**select s.first\_name, b.batch\_name**

**from students s**

**join batches b**

**using(batch\_id);**