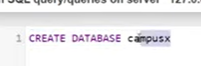




Data base creation



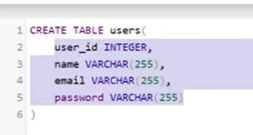
Sql commands caps









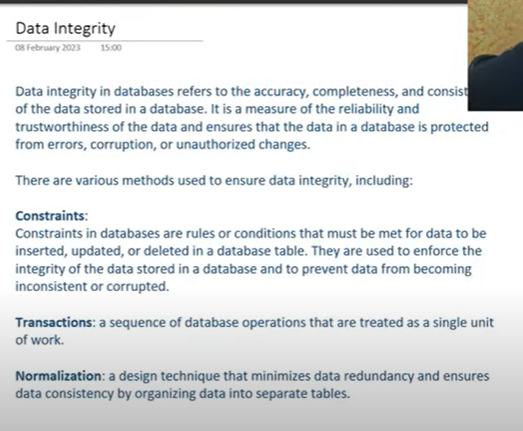


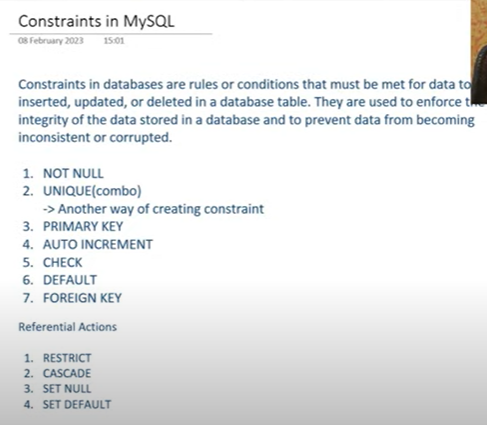


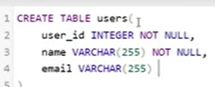
Clears data

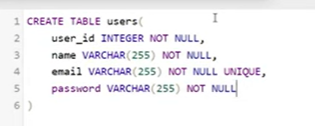


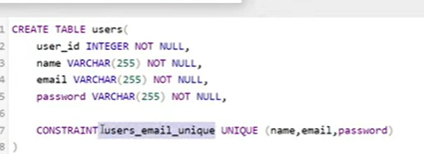
Delete complete table



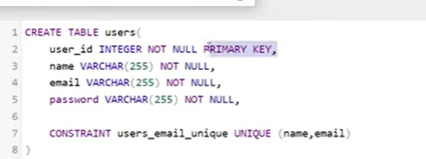


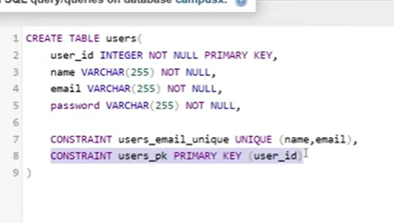


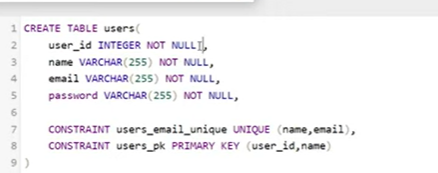




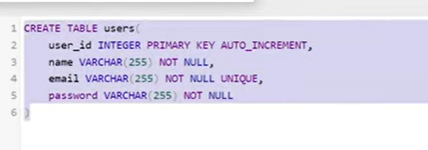
One or more column can be made as primary key

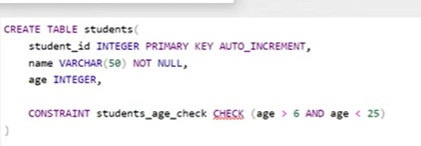


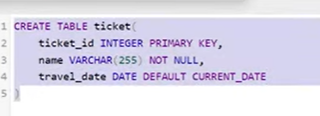




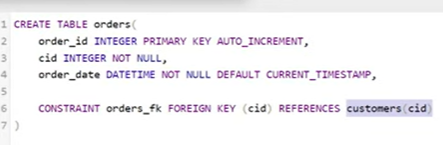
Either way we can create constraints

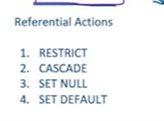






Foreign key



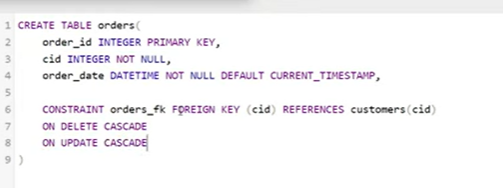


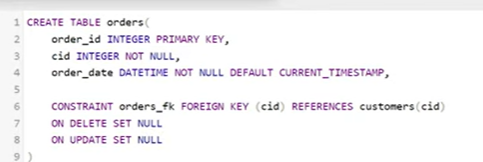
Restrict: will not allow to delete

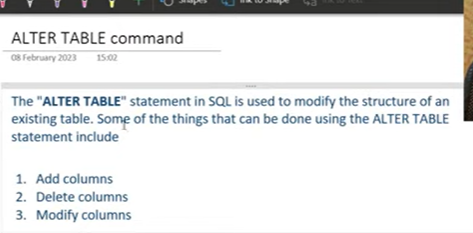
Cascade: automatically all get deleted in another table

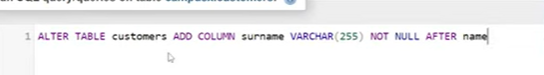
Set null

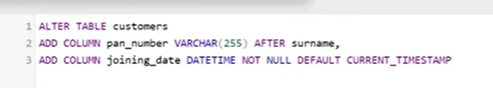
Set default

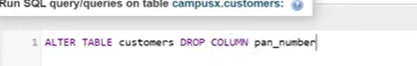


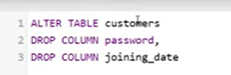












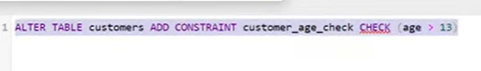


By using alter table: we can delete or create constraints



Edit cant be done in mysql







# 

# - SQL DML Commands

# 

# 

# Database.table

# 

# 

# Values in order

# 

# 

# Filter columns

# 

# Alias – remaining

# 

# 

# Creating expression using col

# 

# 

# Constants

# to create a new column with constant value

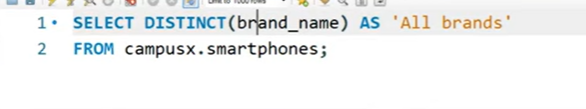
# 

# Here type as new column with value smartphone

# Distinct unique values from column

# 

To get distict values in column



# Distinct combos

# 

# Filtering rows using where

# 

# 

# 

# 

# 

# 

# Order of execution

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Sql functions 2 types

# Builtin

# User defined

# Builtin 2 types

# Scalar functions- round fun(2.55 to 2),abs,lower,upper

# Aggregate function -avg,sum,min,max- single value for entire column

# 

# 

# 

# 

# 

# 

# 

# 

# In scalar func each value generates another value

# 

# 

# Round(percentage,2)

# 

# SQL Grouping + Sorting

# 

# 

# Samsus is one of brand name

# 

# Screen size big

# Sorn based on screen

# 

# 

# 

# Top1

# 

# 

# 

# Camaras two types front and rare

# 

# Pixel per

# 

# 

# Ppi calculate – then sort based on ppi

# 

# 

# 

# LIMIT 5,2

# Means leave till 5 rows take from 6h one and 2 rows

# Second lowest capacity

# 

# 

# 1,Select \* from campus.smartphones

# Where brand=”apple”

# Step2 : Select brand,rating from campus.smartphones

# Where brand=”apple”

# Step3

# Select brand,rating from campus.smartphones

# Where brand=”apple” order by rating ASC LIMIT 1

# 

# 

# Sort based on brand and then rating desc

# 

# 

# 

# Groupby along with min,max,count

# Groupby on categorical column

# Select \* from table group by brand

# Step 2: Select brand from table group by brand

# -Name of brand should come

# Step2:

# Select brand, avg(price),max(rating),avg(screen) from table group by brand

# 

# Further came modified

# Ordering based on num\_phones

# 5 top selling brands

# 

# 

# We have column – have nfc- yes or no

# 

# 

# 

# Group by multiple columns

# 

# 

# Group by both brand and processor

# Step1:Select brand, processor from table

# Group by brand, processor

# Step2 : Select brand, processor,count(\*) as count\_model

# Avg(resolution) as cam\_resolution from table

# Group by brand, processor

# 

# 

# 

# Max price of each brand

# Group by brand

# Step1:

# 1-Select \* from table group by brand

# 2-select brand from table group by brand

# 3-select brand,max(price) from table group by brand

# 3- select brand,max(price) as max\_price from table group by brand order by max\_price

# 4- select brand,max(price) as max\_price from table group by brand order by desc max\_price limit 5

# 

# 

# 1-Select \* from table group by brand

# 2-select brand from table group by brand

# 3-select brand,min(screen) from table group by brand

# 3- select brand,max(price) as min\_screen from table group by brand order by min\_screen

# 4- select brand,max(price) as max\_price from table group by brand order by desc max\_price limit 1

# 

# 

# Brand should have highest number of NFD and IR

# Nfd – true and false

# IR- true/false values

# 

# 

# 

# 

# 

# Having:

# On groups condition – having

# Before groupby where

# 

# 

# 

# Select brand from table group by brand

# Select count(\*) as count, avg(price)brand from table group by brand

# Select count(\*) as count, avg(price),brand from table group by brand having count > 20

# 

# Top 3 brand with highest avg ram

# Condition :Refresh rate 90hz , fast charging available

# Count of brand should be more than 10

# Group by condi- count, avg

# Select brand from table group by brand

# Select brand from table where refresh >=90 and fast\_charge= True group by brand

# Select brand, avg(ram)as avg\_ram,count(\*)from table where refresh >=90 and fast\_charge= True group by brand having count >10 order by avg\_ram desc Limit 5

# 

# 

# Avg price of phones

# Condi after groupby Lrating >70 and num\_phone>10

# Condition: before groupby 5g enab

# Select brand from table group by brand

# Select brand from table where 5g= True group by brand

# Select brand,avg(rating) as avg\_rate, count(\*) as count from table where 5g= True group by brand

# Select brand,avg(price) as avg\_price, count(\*) as count from table where 5g= True group by brand having count>10 avg\_rate>70

# 

# Sql Joins

# 

# 

# 

# All possible rows from two tables

# 

# 

# Inner join

# 98 per we use inner join

# 

# Join data based on specific condition

# Returns rows which meets specific condition

# Common items from both tables

# 

# 

# Left join:

# 

# All rows from left

# Matching rows from right

# If there are no matching rows from right results still be there with null values in the columns from right table

# 

# 

# 

# All rows from right

# Matching from left

# If no matching rows in left still result contains all rows from right with none values in columns of left

# Left table: first table

# Right table: second table

# 

# Full outer join:

# All matching rows from both left and right

# Also non matching rows left and right

# Fills the column values non where there is no match

# 

# 

# 

# 

# Set operations:

# 

# Union: combine result of 2 select statements

# Remove duplicates

# Union all: keeps duplicates

# Except or minus(-): return distict rows appear only in 1st dataset not in second

# 

# 

# 

# Self join:

# 

# Self:Table join by itself

# A table treated as 2 tables

# Each row compare with others rows in same table

# Compare the salaries of empl who work in same dept

# Customers with same building add

# 

# Emergency contact other user

# We need ans :

# Name: emergency friend name

# Condition to join table

# Emergency id in first table, user\_id from second table

# 

# 

# 

# 

# Joining more than one column:

# 

# Joining more than 2 tables

# 

# All columns coming in joined table

# But we can get only wanted columns

# 

# Find out all profitable orders

# 

# 

# 

# 

# 

# Join tables

# Group by customers

# Count(\*)

# Order by count desc

# Limit 1

# 

# 

# 

# 

# Session 35 - Subqueries in SQL | DSMP 2023

# 

# In sql sub query is query within another query

# Nested query

# Inside query is select

# Outside query may be a select, insert,update or delete

# Subquery is executed first

# Based of its result outside query executed

# 

# We can also do order by on score

# But we can also do by using subquery

# 2 parts

# First we should her rating for each movie

# Highest rating

# Then extract corresponding column

# 

# 

# Above is not dynamic since we used hard coded condition

# cut the above query place here

# 

# 

# What it returns : based on that 3 types

# 

# Scalar: one scalar value like max, min

# Row subquery : multiple rows but one column-vector

# Table: multiple rows and multiple columns- matrix

# 

# Based on execution 2 types

# Independent sub query: no relation between inner and outer query,

# Independently we can get

# Ex:

# 

# Inner query execute independently

# Correlated sub query :

# Inner query depends on outer query

# If we execute inner query separately error will come

# 

# When can subquery used

# 

# 

# 

# We can do 2 ways

# 

# We have budjest and revenue

# Profit = revenue -budject

# 2 ways

# Order by

# Select movie, revenue-budject as profit from table orderby profit desc limit 1

# Now subquery

# Now we need

# 

# Select max(revenue-budject) from movies

# 

# 

# Which one run faster

# Second will be faster – based on index

# Based on time complexity

# Above one is faster …

# 

# Count of above avg movies

# First find avg

# Select avg(rating) from table

# 2. select count(\*) from table where rating >select avg(rating) from table

# 

# Find highest rated movie of 2000

# Select movie from table where rating =(select max(rating) from table were year =2000 ) and year=2000

# 

# 

# First we need to get movies where voting is greater than avg voting

# Select movies from table where voting >(select avg(voting) from table

# Select movies from (Select movies from table where voting >(select avg(voting) from table

# )where rating= select max(rate) from (Select movies from table where voting >(select avg(voting) from table

# )

# 

# 

# This will give single value

# 

# 

# 

# Zomato datasbase

# Users table: user id

# Order table:also has userid

# First we need people who ordered

# 1.Select distinct (user\_id) from order

# 2.Select \* from users where userid NOT IN ()

# 3. Select \* from users where userid NOT IN (Select distinct (user\_id) from order)

# 

# 

# Top directors based on gross income

# 

# 

# We don’t want sum we only need directors

# 

# 

# 

# 

# 

# 

# All movies Avg rating >8.5( we shouldn’t consider movies rating less than 25000)

# 

# Now group by based on star and avg score

# 

# Now we want having score >8.5

# 

# Find all movies of these actors

# Select \* from movies where star IN()

# 

# 

# Multiple columns

# 

# 

# We us in after where when we compare multiple values-(IN,NOT IN)

# If it is single we use =

# 

# 

# 

# Highest rated movie of each genger

# First we want genre and max(rating )

# 

# 

# 

# One movie per zoner is the result

# 

# 

# Gonre avg rating

# Movies greater than their avg rating movies

# Select gonre, avg(rating) from table group by gonre

# 

# Gonre varies

# 

# 

# 

# Find favourite food of each customers

# 

# 

# 

# 

# Where and all we can use sub queries

# 

# Get the percentage of each movie compare to the total number of votes

# Name, percent of votes

# Select name, votes/(select sum(votes) from table)\*100 from table

# 

# 

# 

# 

# \

# 1.Select avg(score) from table

# 2. select genre, avg(score) as gen\_score from table group by genre

# 3. select genre, avg(score) as gen\_score from table group by genre having avg(score)>( select genre, avg(score) as gen\_score from table group by genre)

# 

# INSERT:

# Window functions:

# 

# Type of analytical functions

# Perform value for each row based on conditions

# Specification mention in over () clause – specifies partitioning and ordering

# Partitioning divides the rows into groups based on specific column or expression

# While order – defines in which rows are processed within each group

# 

# Avg marks in branch

# 

# 

# We will get like this by groupby

# Window function works in same way but result diff

# 

# Here new column created where we get avg value – each row will be having avg value

# Row by row result we will get

# Where as in group by we will get 2 results if 2 groups are there (though 100 rows are there)

# Avg(maks)

# Over(condition for partition)- group by

# 

# 

# Over() makes every student will get one row with new column

# 

# If we don’t mention condition inside over(), it generates value based on entire table

# 

# Generates avg for each branch becoz partitioned by branch

# 2.

# 

# 2 windows it generates

# 

# 

# Window func also generates agg value like groupby diff is it generates new window with same number of rows

# 

# We have solved this question using correlated sub querries

# 1.Select \* avg(marks) over (partition by branch) as “branch\_avg) from table

# 2. select \* from (Select \* avg(marks) over (partition by branch) as “branch\_avg) from table) t

# 3. select \* from (Select \* avg(marks) over (partition by branch) as “branch\_avg) from table) where t.marks>t.branch\_avg

# 

# Rank:

# 

# Rank to students based on branch and rank

# 

# Above is rank for entire colg

# Now rank based on branch

# 

# 

# 

# 

# Row number: just assign row number

# 

# 

# 

# 

# Find spend on month per customer

# 

# 

# 

# 

# 

# 

# 

# Name of student with highest mark

# Select \*, first\_value(name), over(order by marks desc) from table

# 

# 

# Marks placed in First\_value – Deepak marks will be displayed

# 

# But last value: will not work in a perfect way, values keeps changing

# For this we need to understand frames

# 

# Frame is window function is subset of rows within partition

# -Window function creats group

# -Frames makes sub groups within groups

# -That determine scope of window function

# -Frame defined by rows and between

# -Rows: specifies how many rows included in the frame reative to current row

# -Rows 3 preceding : means include current row and 3 rows that preceeds it in the

# partition

# between : specifies boundaries of frame

# 

# 

# Partion based on branch

# Sort based on marks desc

# Default sql” unbounded preceding to current row”

# When in 1st row(window funct can only see 1st row)

# 2nd row(1st row and 2nd only can be seen)

# So that’s y min keeps changing

# 

# We need to consider entire data

# 

# 

# 

# 2nd topper in each branch

# 

# 

# 

# 

# Last

# 

# Query seems to be very big

# 

# Instead of wring big clause in over : we can mention that clause as window dows as w

# Mention that w

# 

# 

# 

# 

# Month on month revenue growth

# 

# 

# 

# 

# 

# Each ipl and top bats man

# 

# 

# 

# 

# 

# 

# Virat 50th match , 100th and 200th match how many runs

# 

# 

# 

# 

# 

# 

# 

# 

# Moving avg

# 

# 

# 

# Food : percentage

# Only from restaurant 1

# 

# R\_id=restaurant id

# Step 2 joined 2 tables

# 

# Group by food items, sum(amount)

# 

# 

# 

# 