show databases;

use world;

DELIMITER //

CREATE PROCEDURE get\_ppl\_visited\_country (IN cty varchar(52))

BEGIN

SET @cty = CONCAT("%", cty, "%");

select

person.personID,

person.personname,

city.name,

da.dateArrived,

(select dc.name from country dc where dc.code = city.countrycode) CountryName

from (( person person

Inner join hasvisitedcity da ON person.personID = da.personID)

Inner join city city ON da.cityId = city.id)

where city.countrycode in (select g.code from country g where g.name like @cty);

END //

DELIMITER ;

call get\_ppl\_visited\_country("land");

show databases;

use world;

DELIMITER //

Create function ren\_continent (origional VARCHAR(35)) RETURNS VARCHAR(30) Deterministic

Begin

IF(origional like 'North America' or origional like 'South America') Then

SET actual = "Americas";

ELSEIF origional= "Oceania" Then

SET actual = "Australia";

elseif origional= "Antartica" Then

SET actual = "South Pole";

END IF;

Return(actual);

end //

Select ren\_continent("Oceania");

Select country.continent, country.name, country.population

from country country

where country.population >0

and country.population = (select max(b.population) from country b where b.continent = country.continent)

order by country.continent,country.name,country.population;

4.1.4

show databases;

use world;

Select

city.name City,coun.name Country,coun.population population from

(( country coun

Inner join city city ON city.countrycode = coun.code)

Inner join hasvisitedcity hc ON hc.cityId = city.id )

where

hc.personID in (select age.personID from person age where age.age = (select min(per.age) from person per))

and coun.population = (select min(cou.population) from country cou

where cou.code in (select cy.countrycode from city cy where

cy.id in (select hy.cityid from hasvisitedcity hy where

hy.personID in (select pi.personID from person pi

where pi.age = (select min(per.age) from person per)))));

show databases;

use world;

select district,

case

when district = "Eastern Cape"then sum(population)+1000

when district = "Free State"then sum(population)+2000

when district = "Western Cape"then sum(population)-10000

else sum(population)

end population

from city where countrycode in (select code from country where name like "%south%africa%")group by district;

show databases;

use world;

select name, Indepyear,

case

when Indepyear is null

then "N/A"

when cast(year(curdate()) as signed) - Indepyear < 10

then Concat("New ",GovernmentForm)

when (cast(year(curdate()) as signed) - Indepyear >= 10) AND (cast(year(curdate()) as signed) - Indepyear < 50)

then Concat("Modern ",GovernmentForm)

when (cast(year(curdate()) as signed) - Indepyear >= 50) AND (cast(year(curdate()) as signed) - Indepyear < 100)

then Concat("Early ",GovernmentForm)

when (cast(year(curdate()) as signed) - Indepyear >= 100)

then Concat("Old ",GovernmentForm)

else "old"

end Description

from country

order by Indepyear;

**Answer**

* We can normalise as per below
  + **Student table**
  + **Module table**
  + **Student-Module table**
* **Student table** can have the following columns with unique student id
  + Student Id
  + Student name
  + Dob

|  |  |  |
| --- | --- | --- |
| Student ID\* | student Name | dob |
| 1 | Sean | 2000-01-03 |
| 2 | Bill | 1990-04-03 |
| 3 | Tom | 1973-12-10 |
| 4 | Mary | 1991-04-12 |
| 5 | Joe | 1982-06-29 |

* **Module table** can have the following columns with unique module id
  + Module Id [**Primary Key**]
  + Module Name

|  |  |
| --- | --- |
| moduleID\* | moduleName |
| 100 | Applied Databases |
| 101 | Java Programming |
| 102 | Computer Architecture |
| 104 | Mobile Apps |

* **Student-Module table** can have the following columns with combined key of student id and module id

student ID\*; module ID\* are [**Foreign Key**]

|  |  |
| --- | --- |
| studentID\* | moduleID\* |
| 1 | 100 |
| 2 | 100 |
| 3 | 101 |
| 3 | 104 |
| 4 | 101 |
| 4 | 102 |
| 5 | 100 |
| 5 | 104 |

db.docs.aggregate([ { $match: {"car.engineSize": {$exists:true}}}, {$group: {\_id: "car.reg", Average: {$avg:"$car.engineSize"}}} ]);

db.docs.aggregate([ { $bucket: { groupBy: "$pop", boundaries: [0,50000,100000,150000], default: "other", output:{"counties": {$push: "$name"}} }} ]);

db.docs.aggregate([ {$match: {"pop": {$exists:true}}}, {$addFields: {"pop": {$cond: { if: { $lte: [ "$pop", 100000 ] }, then: "Small County", else: "Big County" }}}} ])