**DATA ANALYSIS- MY SQL**

* CREATION OF DATA BASE MySQL\_project .

create database mysql\_project;

* Use data base mysql\_project.

Use database mysql\_project;

DATA SET – KAGGLE DATA SOURCE (Consolidated Tables)

ANALYSING DATA WITH SQL QUERIES

1. select all employees in department 10 whose salary is greater than 3000. [table: employee]

select \* from employee

where deptno =10 & salary > 3000;

2. The grading of students based on the marks they have obtained is done as follows:

40 to 50 -> Second Class

50 to 60 -> First Class

60 to 80 -> First Class

80 to 100 -> Distinctions

1. How many students have graduated with first class?
2. How many students have obtained distinction? [table: students]

select \*,

case

when marks between 40 and 50 then "second class"

when marks between 50 and 60 then "first class"

when marks between 60 and 80 then "first class"

when marks between 80 and 100 then "distinction"

end as status

from students;

select count(name) from students

where marks between 50 and 80;

select count(name) from students

where marks between 80 and 100;

3. Get a list of city names from station with even ID numbers only. Exclude duplicates from your answer.[table: station]

select distinct city from station

where (id%2)=0;

4. Find the difference between the total number of city entries in the table and the number of distinct city entries in the table. In other words, if N is the number of city entries in station, and N1 is the number of distinct city names in station, write a query to find the value of N-N1 from station.

[table: station]

select count(city)-count(distinct city) from station;

5. Answer the following

a. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates. [Hint: Use RIGHT() / LEFT() methods ]

b. Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.

c. Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

1. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates. [table: station]

select city from station

where city like "a%"

or city like "e%"

or city like "i%"

or city like "o%"

or city like "u%";

select distinct(city) from station

where city like "a%a"

or city like "e%e"

or city like "i%i"

or city like "o%o"

or city like "u%u";

select distinct(city) from station

where city not like "a%"

or city not like "e%"

or city not like "i%"

or city not like "o%"

or city not like "u%";

select distinct(city) from station

where city not like "a%a"

or city not like "e%e"

or city not like "i%i"

or city not like "o%o"

or city not like "u%u";

6. Write a query that prints a list of employee names having a salary greater than $2000 per month who have been employed for less than 36 months. Sort your result by descending order of salary. [table: emp]

select concat(first\_name," ",last\_name) as name from emp

where salary > 2000 and (timestampdiff(month,hire\_date,now()))<36

order by salary desc;

7. How much money does the company spend every month on salaries for each department? [table: employee]

Expected Result

----------------------

+--------+--------------+

| deptno | total\_salary |

+--------+--------------+

| 10 | 20700.00 |

| 20 | 12300.00 |

| 30 | 1675.00 |

+--------+--------------+

3 rows in set (0.002 sec)

select deptno,sum(salary) from employee

group by deptno;

SET2

1. Write a stored procedure that accepts the month and year as inputs and prints the ordernumber, orderdate and status of the orders placed in that month.

***Example***: call order\_status(2005, 11);

create procedure order\_status(in e\_year int, in e\_month int)

begin

select ordernumber,orderdate,status from orders

where month(orderdate)=e\_month and year(orderdate) = e\_year;

end \\

call order\_status(2003,11);

2. a. Write function that takes the customernumber as input and returns the purchase\_status based on the following criteria . [table:Payments]

if the total purchase amount for the customer is < 25000 status = Silver, amount between 25000 and 50000, status = Gold

if amount > 50000 Platinum

delimiter \\

create procedure pay\_status(in p\_customernumber int,out p\_status varchar(100))

begin

declare var decimal;

select sum(amount) into var from payments

where customernumber=p\_customernumber

group by customernumber;

if var<25000 then set p\_status="silver";

elseif var between 25000 and 50000 then set p\_status= "gold";

else set p\_status= "platinum";

end if;

end\\

call pay\_status(101,@abc);

select @abc as purchase\_status;

-- user defined function

delimiter \\

create function pay\_stattus(xyz int)

returns varchar(100)

deterministic

begin

declare var decimal;

declare p\_status varchar(100);

set var = (select sum(amount) from payments

where customernumber=xyz

group by customernumber);

if var<25000 then set p\_status="silver";

elseif var between 25000 and 50000 then set p\_status= "gold";

else set p\_status= "platinum";

end if;

return p\_status;

end \\

select customernumber,pay\_stattus(100)

from payments;

b. Write a query that displays customerNumber, customername and purchase\_status from customers table.

select customernumber ,customername,

case

when creditlimit<25000 then "silver"

when creditlimit between 25000 and 50000 then "gold"

when creditlimit >50000 then "platinum"

end as purchase\_status

from customers;

3. Replicate the functionality of 'on delete cascade' and 'on update cascade' using triggers on movies and rentals tables. Note: Both tables - movies and rentals - don't have primary or foreign keys. Use only triggers to implement the above.

delimiter \\

create trigger movie\_rentals

after delete

on movies for each row

begin

update rentals

set movieid = null

where movieid=old.id;

end \\

delimiter \\

create trigger movie\_rentals\_update

after update

on movies for each row

begin

update rentals

set movieid=new.id

where movieid=old.id;

end \\

update movies

set id= 11

where id=10;

4. Select the first name of the employee who gets the third highest salary. [table: employee]

select fname from employee

order by salary desc

limit 2,1;

5. Assign a rank to each employee based on their salary. The person having the highest salary has rank 1. [table: employee]

select empid , salary,

rank()over(order by salary desc) as emp\_rank

from employee;

based the scenarios mysql queries are written and the data set is analysed .