**Name: \_\_Michael Adams\_\_\_\_\_\_\_\_\_**

**Part 2 –Queries with Join**

This is where you start to turn things in. To turn in your assignment, delete the preceding pages and start with this one. Make sure to put your name up top.

Construct SQL to answer each of the following questions. For each query, paste your SQL code below and also paste a screenshot of the first 10 or so rows the query gives you.

Give an answer to the question based only on what you see in your MySQL database. (Do not use Excel or other tools to answer these questions; since we are trying to learn MySQL here, you should do these exercises in MySQL.)

You will eventually submit this document as part of your week’s homework.

After you have completed your first draft of this document, you will use the online quiz to verify your answers. The online quiz uses the same numbering system given here, so you can quickly find the question you need.

You are welcome to update your homework document with results from the online quiz, and submit the homework document after you’ve checked your homework online.

Write SQL queries to answer the following questions:

1. [SQL Week 2 Question 200-001] Print the first name, last name, ticket ID, ticket description, and duration of the employees who are assigned with a ticket(s). Sort them first alphabetically by first name (starting with letter A at the top) and then ascending by Duration.

SELECT

FirstName, LastName, idTickets, Description, Duration

FROM

tickets

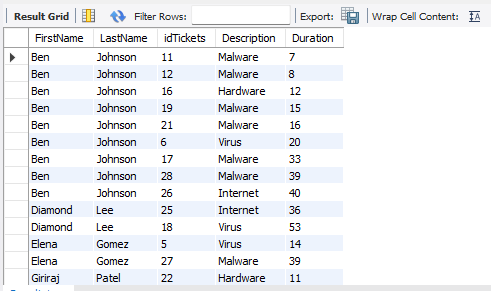
JOIN

employee ON tickets.Employee\_idEmployee = employee.idEmployee

WHERE

tickets.Employee\_idEmployee IS NOT NULL

ORDER BY FirstName , Duration ASC;



1. [SQL Week 2 Question 200-002] We want to get a workload by employee printout. For every employee, print the first name, last name, all ticket ID numbers assigned to that employee, the ticket description, and ticket duration of our records. Include only tickets which have been assigned to an employee. Include all employees, even if that person doesn’t have any tickets in the current database. Sort them first alphabetically by first name (starting with letter A at the top) and then ascending by ticket duration.

SELECT

employee.FirstName,

employee.LastName,

tickets.idTickets,

tickets.Description,

tickets.Duration

FROM

employee

LEFT JOIN

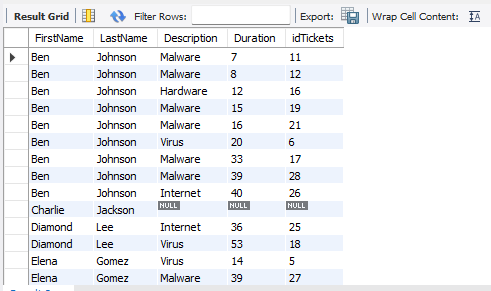
tickets ON employee.idEmployee = tickets.Employee\_idEmployee

WHERE

tickets.Employee\_idEmployee IS NOT NULL

OR tickets.Employee\_idEmployee IS NULL

ORDER BY employee.FirstName , tickets.Duration ASC;



1. [SQL Week 2 Question 200-003] We want to get an idea of how many tickets we have and what our issues are. Print the ticket ID number, ticket description, ticket priority, ticket status, and if the information is available, employee first name assigned to it for our records. Include all tickets regardless of whether they have been assigned to an employee or not. Sort it alphabetically by ticket status, and then numerically by ticket ID, with lower ticket IDs on top.

SELECT t.idTickets AS 'Ticket ID',

t.Description,

t.Priority,

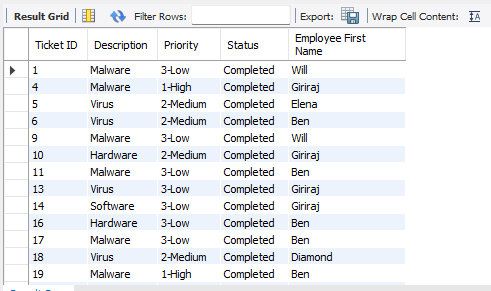
t.Status,

e.FirstName AS 'Employee First Name'

FROM tickets t

LEFT JOIN employee e ON t.Employee\_idEmployee = e.idEmployee

ORDER BY t.Status, t.idTickets ASC;



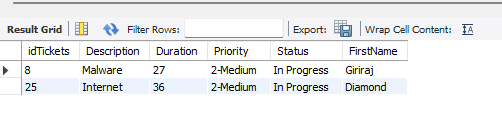
1. [SQL Week 2 Question 200-004] Management wants to pay special attention to calls which are in progress and for which the duration is currently 20 minutes or longer (include calls of exactly 20 minutes duration). Print the ticket ID number, description, duration, priority, status, and the employee first name for all of these calls.

**SELECT tickets.idTickets, tickets.Description, tickets.Duration, tickets.Priority, tickets.Status, employee.FirstName**

**FROM tickets**

**JOIN employee ON tickets.Employee\_idEmployee = employee.idEmployee**

**WHERE tickets.Status = 'In Progress' AND tickets.Duration >= 20;**



1. [SQL Week 2 Question 200-005] You suspect there has been a malware breach at the Maryland facility. They’re not sure if it’s an inside job (involving one of your employees) or an outside job (involving an outside attack.) Make a listing of all high priority helpdesk tickets which have either been identified as malware, or which were taken by an employee whose location is in the Maryland facility. Print the description, duration, ticket ID number, ticket status, employee first name, employee location, and priority of ticket. Sort it alphabetically by description, then by duration (longest duration on top), then ascending by ticket ID number.

SELECT t.description, t.duration, t.idtickets, t.status, e.firstname, e.location, t.priority

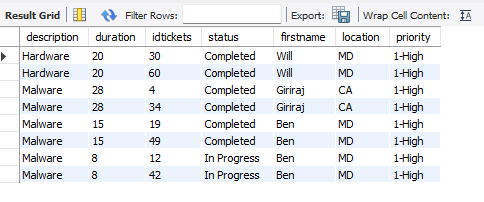
FROM employee e, tickets t

WHERE e.idemployee=t.employee\_idemployee

AND t.priority = '1-High' AND (t.description = 'Malware'

OR e.location = 'MD')

ORDER BY t.description, t.duration DESC, t.idtickets;



1. [SQL Week 2 Question 200-006] We want to see the total duration for completed helpdesk calls by employee. Only include employees who have completed helpdesk tickets; if an employee doesn’t have any completed tickets, their name should not show up in this report. Generate a report for which the
   * First column contains the employee’s first and last name (don’t use two columns for this – put the first and last name together)
   * Second column contains the total duration of all completed tickets by that employee. If an employee had more than one ticket, add the durations together.
   * Report is sorted so the largest total duration is on the top, and then alphabetically by employee name (sort it using the “First Last” combination, so that “Mickey Mouse” would come before “Nancy Mouse”.)
   * Only include completed tickets; don’t include those in progress or not yet started.
   * *Hint: look up the CONCAT and CONCAT\_WS operators in W3 Schools’ MySQL. You can find it on the left menu under SQL References-> MySQL Functions, or you can simply click on the magnifying glass search icon on the top right, and type in “CONCAT” to see.*

For example, assume your initial data was as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| Employee First Name | Employee Last Name | Ticket status | Ticket Duration |
| Mickey | Mouse | Completed | 5 |
| Mickey | Mouse | Completed | 10 |
| Donald | Duck | In Progress | 15 |
| Donald | Duck | Completed | 20 |
| Donald | Duck | Completed | 25 |
| Donald | Duck | Completed | 30 |

Your report would contain the following information:

|  |  |
| --- | --- |
| Employee Name | Duration of completed tickets |
| Donald Duck | 75 |
| Mickey Mouse | 15 |

Note that Mickey Mouse has two calls here, both of which were completed, so their duration is 5 + 10 = 15. Donald Duck has four calls here, but only three of them are completed, so his duration is 20 + 25 + 30 = 75 minutes; Donald’s 15 minute call is still in progress and is not counted here. And Donald’s 75 minutes is the largest number of minutes, so it goes on top.

SELECT Priority, Description,

AVG(Duration) AS AverageDuration

FROM tickets

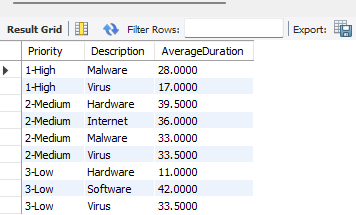
JOIN employee ON tickets.Employee\_idEmployee = employee.idEmployee

WHERE employee.Location = "CA"

AND tickets.Status IN ('Completed', 'In Progress')

GROUP BY Priority, Description

ORDER BY Priority ASC, Description ASC;



1. [SQL Week 2 Question 200-007] Do the same as Problem 6 above, except in addition to the sum of the ticket durations, also include an additional column to give us the number of completed tickets as well. Now sort your database so the largest number of completed tickets is on the top.

Here's a sample of the output for Mr. Duck and Mr. Mouse.

|  |  |  |
| --- | --- | --- |
| Employee Name | Duration of completed tickets | Total number of completed tickets |
| Donald Duck | 75 | 3 |
| Mickey Mouse | 15 | 2 |

SELECT

CONCAT(employee.FirstName, ' ', employee.LastName) AS EmployeeName,

SUM(tickets.Duration) AS TotalDuration,

COUNT(tickets.idTickets) AS NumberOfTickets

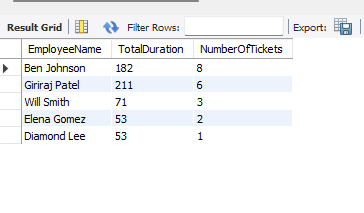
FROM tickets

JOIN employee ON tickets.Employee\_idEmployee = employee.idEmployee

WHERE tickets.Status = 'Completed'

GROUP BY employee.FirstName, employee.LastName

ORDER BY NumberOfTickets DESC, EmployeeName ASC;



1. [SQL Week 2 Question 200-008] You want to learn more about the average duration of all tickets taken by employees in California. Include all tickets which have been completed or are in progress by employees in California. Sort them alphabetically by priority, and then alphabetically by ticket description. Additionally, group them first by priority (highest on top), then alphabetically by description. So your final report will look something like this (your data may vary):

|  |  |  |
| --- | --- | --- |
| Priority | Description | Average duration of tickets |
| 1-High | Malware | 99 |
| 1-High | Virus | 12 |
| 2-Medium | Internet | 5 |
| 2-Medium | Malware | 8 |
| 2-Medium | Virus | 3 etc. |

SELECT Priority, Description, AVG(Duration) AS AverageDuration

FROM tickets

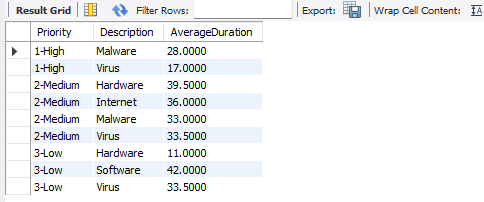
JOIN employee ON tickets.Employee\_idEmployee = employee.idEmployee

WHERE employee.Location = 'CA'

AND tickets.Status IN ('Completed' , 'In Progress')

GROUP BY Priority , Description

ORDER BY Priority ASC , Description ASC;



1. [SQL Week 2 Question 200-009] Your management wants to know what the different types of tickets are. Write a query that uniquely identifies ticket type based on the ticket description.

SELECT DISTINCT Description AS idTickets

FROM tickets

ORDER BY idTickets ASC;

