SQL – linked in

Relational database

Data- datum = piece of info, text, images photos, any data

Data base- collection of data, organized in different ways usually in tables.

Tables- rows\*columns

Each row represents single piece of data called record.

Each column represents a specific attribute of data ex: name or address.

SQL – structured query language = sequel

Is used to analyze database.

Select statement allows us to specify the columns in tables.

Relational Database MANAGEMENT SYSTEM:

SOFTWARE system used to compose sql statements is called rdbms

RdbMs allows us an area to compose sql statements.

SQL STATEMENGT ;

A screenshot of a computer

Description automatically generated

the select clause:

The from clause:

The order by clause:

We used dB browser in this course, because it is a free open source which means it is open to public. Can be used to create edit design databases

To get data from DBMS we should first understand question properly and then covert into query

A screenshot of a computer

Description automatically generated

WRITING A QUERY

-- (OR) /\* \*/ = THIS IS A COMMENT

/\*

THIS IS A COMMENT

CREATED : BY WALTER SHIELDS

WHEN :MM/DD/YY

DESCRIPTION : THIS IS A STRUCTURE OF A BASIC QUERY

\*/

QUERY 1)

/\*

CREATED BY ; GANGADHAR

DESCRIPTION : THIS QUERY DISPLAYS ALL CUSTOMERS FIRST AND LAST NAME ALONG WITH EMAIL

\*/

SELECT

FirstName,

LastName,

Email

FROM

Customer

ALIASES : ALLOWS US TO RENAME OUR COLUMS

SELECT

FirstName AS [Customer FIRST Name],

Last Name,

Email

FROM

Customer

OUT OUT : FIRST NAME IN THE TABLE HAS BEEN CHANGED INTO CUSTOMERS FRIST NAME

/\*

CREATED BY ; GANGADHAR

DESCRIPTION : THIS QUERY DISPLAYS ALL CUSTOMERS FIRST AND LAST NAME ALONG WITH EMAIL

\*/

SELECT

FirstName AS[Customer FIRST Name],

LastName AS[Customer last name],

Email AS EMAIL

FROM

Customer

ORDER BY

FirstName ASC,

LastName DESC

LIMIT 10

/\*

WHERE CLAUSE

OPERATORS

ARITHAMATIC,COMPARISION,LOGICAL

[+-/\*] [= <><=,>=] [AND, OR , IN, LIKE, BETWEEN

\*/

FOR A COMOANY THAT REQUIRES DATA FROM INNVOICE BETWEEN 1 AND 3.96$

SELECT

InvoiceDate,

BillingAddress,

BillingCity,

total

FROM

Invoice

WHERE

total BETWEEN 1.98 AND 3.96

ORDER BY

InvoiceDate

JOIN COMMAND : A COMMAND that combines fields from two or more tables of a relational database

query

--join {when data matches completely}

--inner join : only represents the common values in both the table , it represents overlapping section of Ven diagram

SELECT

\*

FROM

Invoice

INNER JOIN

Customer

on

invoice.CustomerId=Customer.CustomerId

-- table1.columnx1= table2.columnx1

ORDER by

Customer.CustomerId

--join {when data matches completely}

--inner join : only represents the common values in both the table , it represents overlapping section of Ven diagram

SELECT

c.FirstName,

c.LastName,

i.CustomerId,

i.InvoiceDate,

i.total

FROM

Invoice as i

INNER JOIN

Customer as c

on

i.CustomerId = C.CustomerId

-- table1.columnx1= table2.columnx1

ORDER by

C.CustomerId

Left outer join select data from the left table {circle [in a ven diagram]},

Right join is a mirror image of left join , right join doesn’t support in sql lite

Query

Joining more than 2 tables

SELECT

e.FirstName,

e.LastName,

e.EmployeeId,

c.FirstName,

c.LastName,

c.SupportRepId,

i.CustomerId,

i.total

FROM

Invoice as i

INNER JOIN

Customer as c

ON

I.CustomerId=C.CustomerId

INNER JOIN

Employee as e

ON

c.SupportRepId = e.EmployeeId

ORDER BY

i.total DESC

limit 10

query

FUNCTIONS

-- || :- helps us in joining statements

SELECT

FirstName,

LastName,

Address,

FirstName||LastName

FROM

Customer

WHERE

Country='USA'

-- || ' ' || :- helps us giving the space inbetween joined statements

SELECT

FirstName,

LastName,

Address,

FirstName|| ' ' ||LastName

FROM

Customer

WHERE

Country='USA'

SELECT

FirstName,

LastName,

Address,

FirstName|| ' ' ||LastName|| ' ' ||Address

FROM

Customer

WHERE

Country='USA'

-- || ' , ' || :- HELPS US TO GET A , BETWEEN

SELECT

FirstName,

LastName,

Address,

FirstName|| ' ' ||LastName|| ' ' ||Address|| ' , ' ||City

FROM

Customer

WHERE

Country='USA'

-- helps us add more info

SELECT

FirstName,

LastName,

Address,

FirstName|| ' ' ||LastName|| ' ' ||Address|| ' , ' ||City|| ' ' ||State|| ' ' ||PostalCode as [Mailling Adress]

FROM

Customer

WHERE

Country='USA'

Aggregate function

-- aggregate function turnd s rsnge of numbers into single data

SELECT

sum(total)as[total Sales],

avg(total) as [average sale},

max

min

count

FROM

Invoice

TRUNCATE : TO SHORTEN OR REDUCE TEXT

SELECT

FirstName,

LastName,

Address,

FirstName|| ' ' ||LastName|| ' ' ||Address|| ' , ' ||City|| ' ' ||State|| ' ' ||PostalCode as [Mailling Adress],

length(PostalCode),

substr(PostalCode,1,5)as[5 Digit PostalCode]

FROM

Customer

WHERE

Country='USA'

* Using upper and lower cases like capital and non capital letter

-- upper() function :- converts a string pf text to upeercase[

SELECT

FirstName,

LastName,

Address,

FirstName|| ' ' ||LastName|| ' ' ||Address|| ' , ' ||City|| ' ' ||State|| ' ' ||PostalCode as [Mailling Adress],

length(PostalCode),

substr(PostalCode,1,5)as[5 Digit PostalCode],

upper(FirstName)as [FirstName all caps],

lower(LastName)as[ LastName all lower]

FROM

Customer

WHERE

Country='USA'

DATE FUNCTIONS

-- datefunction striftime

SELECT

FirstName,

LastName,

BirthDate,

strftime('%Y-%m-%d',Birthdate)as[BirthDate no timecode]

FROM

Customer

WHERE

Country='USA'

AGGREGATE FUNCTIONS|WHAT ARE ALL TIME GLOBAL SALES

SELECT

sum(TOTAL)AS [total SALES],

avg(TOTAL) AS [AVG SALES],

MAX(TOTAL)AS[MAX SALES],

COUNT(\*)AS[SALES count]

FROM

Invoice

NESTING FUNCTIONS : USED TO ROUND OFF NUMBERS

SELECT

sum(TOTAL)AS [total SALES],

round(avg(TOTAL) ,2) AS [AVG SALES],

MAX(TOTAL)AS[MAX SALES],

COUNT(\*)AS[SALES count]

FROM

Invoice

**Use full feature of aggregate function:**

GROUP BY

SELECT

BillingCity,

round(avg(total),2)

FROM

Invoice

group by

BillingCity

order by

BillingCity

What are the average invoice totals by city for only the cities that start with l

SELECT

BillingCity,

round(avg(total),2)

FROM

Invoice

WHERE

BillingCity like 'L%'

group by

BillingCity

order by

BillingCity

What are the average invoice totals greater than 5$

A HAVING clause is like a WHERE clause, but applies only to groups as a whole (that is, to the rows in the result set representing groups), whereas the WHERE clause applies to individual rows

SELECT

BillingCity,

round(avg(total),2)as [avg total]

FROM

Invoice

group by

BillingCity

HAVING

avg(total)>5

ORDER by

BillingCity ASC

What are the average invoice totals greater than 5$ for cities starting with letter b

SELECT

BillingCity,

round(avg(total),2)as [avg total]

FROM

Invoice

WHERE

BillingCity like "B%"

group by

BillingCity

HAVING

avg(total)>5

ORDER by

BillingCity

What are the average invoice totals by biling country and biling city

SELECT

BillingCity,

BillingCountry,

round(avg(total),2)as [avg total]

FROM

Invoice

group by

BillingCity,BillingCountry

ORDER by

BillingCountry

**NESTED QUERY : WHEN A QUERY IS WRAPPED/WRITEEN IN SIDE OF ANOTHER QUERY**