Group 52

CS6400 Report Phase 2

Abstract Code & SQL

Molly Albertson, Gopi Krishna Bandi, Tomas Ordonez, Justin Swiderski, Minyan Zhang

3-22-2021

Contents

Changes to Phase 1

On the updated EER:

- We removed the Store N:M Product relationship as it was not essential.
- From the weak entity Sale, we removed the composite attribute "Sales Information" and the calculated attribute "Total Amount" as they were not essential.
- We updated the cardinality of Store 1:1 Child Care Limit and updated the Child Care Limit entity name: Store N:1 Limit.
- From Holiday we removed the attribute "Holiday Date" as it was not essential.
- Stores don't run Campaigns so we removed the relationship between Store and Campaign.
- From Campaign we removed the attribute "Campaign Date" as it was not essential.
- From Date we changed the composite attribute Date (composed of Day, Month, Year) to a simple attribute Date
- We removed the Store 1:N Holiday relationship as we saw that all stores have all the holidays, like all stores have all products, so it was not essential.
- We updated the cardinality Date N:1 Campaign to Date 1:N Campaign.
- We updated the cardinality Date N:M Holiday to Date 1:N Holiday.

Abstract Code w/SQL

Main Menu Report

Abstract Code

- User Enters the application to view the statistics on the **Dashboard**.
- When user clicks ***View Statistics***, the Dashboard table should populate with information from the following query which includes the count of stores, count of stores that offer food (have a restaurant, a snack bar, or both), count of stores offering childcare, count of products, and count of distinct advertising campaigns.
- Find all Store; Count all Store; Count the Store whose Restaurant Flag is True, or Snack Bar Flag is True; Count the Store whose Child Care Flag is True.

SELECT count(DISTINCT Store_Number) FROM Store;
SELECT count(DISTINCT Store_Number) FROM Store WHERE Restaurant_Flag = 1 OR SnackBar_Flag = 1;
SELECT count(DISTINCT Store_Number) FROM Store WHERE Childcare_Flag = 1;

- Find all Product; Count all Product.

```
SELECT count(DISTINCT PID) FROM Product;
```

- Find all Campaign; Count all Campaign.

```
SELECT count(DISTINCT `Description`) FROM Campaign;
```

Holiday Interface

Abstract Code

User Enters the application **Dashboard** and clicks on the **Holiday** Menu. The system then runs **View/Add Holiday** task:

- User clicks on ***Add Holidays*** button.
- *Date selection* Form is enabled to select the date.

```
SELECT DISTINCT 'Date' FROM Date;
```

- *Holiday* Text Box is enabled to enter the holiday name and date.
- Throw an error POP UP when the holiday is already in the database.
- ***Submit*** button must be clicked to submit changes to the database.

```
INSERT INTO Holiday(Holiday_Name, `Date`) VALUES('$Holiday_Name', '$Date');
```

For Viewing Holidays, user clicks on ***View Holidays*** button

- List of Holidays is displayed

```
SELECT DISTINCT Holiday_Name, 'Date' FROM Holiday;
```

Update City Population

Abstract Code

User Enters the application **Dashboard** and clicks on the **Update City Population** Menu, then the system runs the **Update** task:

- *Update City* drop down form is enabled to select the city.

```
SELECT City_Name FROM City;
```

- *Select State* drop down form is enabled to select the state in which the city is.

```
SELECT distinct State.State_Name FROM State JOIN City on City.State_Name = State.State_Name
where City.City_Name='$City';
```

- *Population* Text Box is enabled to enter the population.
- Throw an error POP UP when the population is not an integer.
- ***Submit*** button must be clicked to submit changes to the database.

```
UPDATE City SET Population = '$Population' WHERE City.City_Name = '$City' AND City.State = '$State';
```

Report 1 Category Report

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 1 Category Report*** button, then the system runs **Report 1** task:

- For each Category (including those without products), combine with Product to Display the category name, total number of products in that category, the minimum regular retail price, the average regular retail price, and the maximum regular retail price of all the products in that category, sorted by category name ascending.

```
SELECT C.Category_Name, count(DISTINCT P.PID), min(P.Retail_Price), avg(P.Retail_Price),
max(P.Retail_Price)
FROM Category C LEFT OUTER JOIN Product P on C.PID = P.PID GROUP BY C.Category_Name
ORDER BY C.Category_Name ASC;
```

Report 2 Actual versus Predicted Revenue for Couches and Sofas

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 2 Actual versus Predicted Revenue for Couches and Sofas*** button, then the system runs **Report 2** task:

- Combine Product and Category to get the product ID, Product Name and Retail Price for Couches and Sofas Category.
- Combine the above results with Sale to get Total units Sold.
- Combine the above results with Discount and Date to get Discount Price by Dates.
- Combine Sale, Discount and Date to get the total units sold at a discount, the total units sold at retail price and the actual revenue.
- Calculate

- Display records for which predicted revenue differences greater than $5000 (positive or negative) by filtering the records.
- Sort records by predicted revenue differences in descending order.

```
Select P.PID, P.Product_Name, P.Retail_Price,

sum(Sa.Quantity) as 'Total Units',

sum(case when D.Discount_Price is null then Sa.Quantity else 0 end ) as 'Retail Units',

sum(case when D.Discount_Price is null then 0 else Sa.Quantity end) as 'Discount Units',

sum(case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then D.Discount_Price else P.Retail_price
end *Sa.Quantity)as 'Actual Revenue',

sum(case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then P.Retail_price else P.Retail_price end
*case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then Sa.Quantity*0.75 else Sa.Quantity end) as
'Predicted Revenue',

sum(case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then D.Discount_Price else P.Retail_price
end *Sa.Quantity)-sum(case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then P.Retail_price else
P.Retail_price end *case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then Sa.Quantity*0.75 else
Sa.Quantity end) as 'Difference'

FROM Product P

join Category C

on C.PID=P.PID

join

Sale Sa

on P.PID=Sa.PID

join Date DT

on Sa.`Date`=DT.`Date`

left outer join Discount D

on Sa.PID=D.PID

and Sa.`Date`=D.Discount_Date

Where C.Category_Name='Couches and Sofas'

group by P.PID,P.Product_Name,P.Retail_Price

having `Difference`>5000

order by `Difference` desc;
```

Report 3 Store Revenue by Year by State

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 3 Store Revenue by Year by State*** button, then the system runs **Report 3** task:

- Select *State Name* from the drop down on the dashboard which is retrieved from State.

```
SELECT DISTINCT State_Name FROM State;
```

- Connect the Store entity with State via the City to get the store ID, store address, city name.
- Combine the above result with Sale and Date to get sales year.
- Combine with Product and Discount to get total revenue (quantity * price) with price either being retail price or discount price based on whether discount was given.
- Display the result by sorting by ascending Year and revenue descending.

```
select S.Store_Number,S.Street_Address,S.City_Name,year(Sa.`Date`) as 'Year',
sum(case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then D.Discount_Price else P.Retail_price end*Sa.Quantity) as 'Total Revenue'
from Store S
join City C on  S.City_Name=C.City_Name
and C.State_Name=S.State_Name
join State st on St.State_Name=C.State_Name
and S.State_Name=St.State_Name
join Sale Sa on S.Store_Number=Sa.Store_number
join `Date` Dt on Dt.`Date`=Sa.`Date`
join Product P on Sa.PID=P.PID
left outer join Discount D
on Sa.PID=D.PID
and Sa.`Date`=D.Discount_Date
where St.State_Name= '$State'
group by S.Store_Number,S.Street_Address,S.City_Name,`Year`
order by `Year` asc,`Total Revenue` desc;
```

Report 4 Outdoor Furniture on Groundhog Day

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 4 Outdoor Furniture on Groundhog Day*** button is clicked, then the system runs **Report 4** task:

- UseDate to find all Year.
- Combine the above results with Sale, Category, Product to find the total units sold in the outdoor category for each Year.
- Find the average number of sold per day (divide the total units sold by 365).
- For each Year, find the Date of the Groundhog Day and combines this Date with Sale, Category, Product to find the total units sold in outdoor category.
- Display the report by year ascending.

```
select distinct year(Sa.`Date`) as 'Year',sum(Sa.Quantity) as 'Total Units',sum(Sa.Quantity)/365 as 'Avg Units Per Day',

sum(case when month(Sa.`Date`)=2 and day(Sa.`date`)=2 then Sa.Quantity Else 0 end) as 'Units Sold on GH Day'

from Sale Sa

join `Date` Dt on

Dt.`Date`=Sa.`Date`

join Product P on

Sa.PID=P.PID

join Category Ca on

P.PID=Ca.PID

where Ca.Category_Name='Outdoor Furniture'

group by `Year`;
```

Report 5 State with Highest Volume for each Category

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 5 State with Highest Volume for each Category*** button, then the system runs **Report 5** task:

- Choose *Year* from the drop down and *Month* from the Month Drop down which is populated by combining Date and Sale.

```
SELECT DISTINCT YEAR(Sa.'Date') FROM Sale Sa JOIN `Date` Dt ON Sa.'Date' = Dt.'Date';
```

```
SELECT DISTINCT MONTH(Sa.'Date') FROM Sale Sa JOIN `Date` Dt ON Sa.'Date' = Dt.'Date'
WHERE YEAR(Sa.'Date') = '$year';
```

- From Category select Category Name.
- Combine Category with Store Via Product.
- Combine with City and State to get State Name for each Category Name**.**
- Combine to get the state name that sold the highest number of units in each category Name, and the number of units that were sold by stores in that state.

```
select C.Category,B.State_Name,C.Number_Of_Units from (
select distinct A.Category as `Category`, max(Quant) over (partition by A.`Category`) as `Number_Of_Units`
from (
select Ca.Category_Name as Category, St.State_Name as State_Nm, sum(Sa.Quantity) as Quant
from Store S
join Sale Sa
on S.Store_Number=Sa.Store_Number
Join City C on
S.City_Name=C.City_Name
and C.State_Name=S.State_Name
join State st
on St.State_Name=C.State_Name
and S.State_Name=St.State_Name
join Product P on
Sa.PID=P.PID
join Category Ca on
```

```
P.PID=Ca.PID
join `Date` Dt on
Dt.`Date`=Sa.`Date`
where year(Sa.`Date`)='$Year'
and Month(Sa.`Date`)='$Date'
group by 1,2)A)C
join (
select Ca.Category_Name as Category, St.State_Name as `State_Name`, sum(Sa.Quantity) as Quantity
from Store S
join Sale Sa
on S.Store_Number=Sa.Store_Number
Join City C on
S.City_Name=C.City_Name
and C.State_Name=S.State_Name
join State st
on St.State_Name=C.State_Name
and S.State_Name=St.State_Name
join Product P on
Sa.PID=P.PID
join Category Ca on
P.PID=Ca.PID
join `Date` Dt on
Dt.`Date`=Sa.`Date`
where year(Sa.`Date`)='$Year'
and Month(Sa.`Date`)='$Date'
group by 1,2)B
on C.Category=B.Category
and C.Number_Of_Units=B.Quantity;
```

Report 6 Revenue by Population

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 6 Revenue by Population*** button, then the system runs **Report 6** task:

- Select population from City and display conditionally the category as Small (population <3,700,000), Medium (population >=3,700,000 and <6,700,000), Large (population >=6,700,000 and <9,000,000) and Extra Large (population >=9,000,000) of each city name.
- Combine with Store to get Store Number which can be used to combine with Sale.
- Retrieve Quantity and PID from Sale and combine on PID with Product and Discount to Calculate. Total Revenue (Product ID Price * Quantity). Use Discount price if there is a discount else use Retail Price.
- Aggregate to get Total Revenue by City Name and Year by combining with Date.
- Display Total Revenue broken down on an annual basis with year ascending and city size category ascending order.

```
select distinct year(Sa.`Date`),sum(if(C.population<3700000, case when D.Discount_Price is null then P.Retail_Price else D.Discount_Price end *Sa.Quantity , 0 )) as 'Small',

sum(if( C.population >=3700000 and C.population<6700000,case when D.Discount_Price is null then P.Retail_Price else D.Discount_Price end *Sa.Quantity,0)) as 'Medium',

sum(if( C.population >=6700000 and C.population<9000000,case when D.Discount_Price is null then P.Retail_Price else D.Discount_Price end *Sa.Quantity,0)) as 'Large',

sum(if( C.population >=9000000,case when D.Discount_Price is null then P.Retail_Price else D.Discount_Price end *Sa.Quantity,0)) as 'Extra Large'

from Store S

join City C on

S.City_Name=C.City_Name

and C.State_Name=S.State_Name

join State st

on St.State_Name=C.State_Name

and S.State_Name=St.State_Name

join Sale Sa on

S.Store_Number=Sa.Store_number
```

```
join `Date` Dt on
Dt.`Date`=Sa.`Date`
join Product P on
Sa.PID=P.PID
left outer join Discount D
on Sa.PID=D.PID
and Sa.`Date`=D.Discount_Date
group by 1
order by 1 asc;
```

Report 7 Childcare Sales Volume

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 7 Childcare Sales Volume*** button, then the system runs **Report 7** task:

- Combine Store and Childcare Limit to retrieve store names which have childcare along with childcare time limit values.
- Conditionally format and group retrieved data with stores having No Childcare as “No childcare” as category.
- Retrieve Quantity and PID from Sale and combine on PID with Product and Discount to Calculate Total Sales (Product ID Price * Quantity). Use Discount price if there is a discount else use Retail Price.
- Combine with Date on date of sale to filter the last 12 months of data and aggregate (sum) the revenues by month.
- Display in a tabular format, with row values for each month, and column values based on the grouped childcare time limit values for stores, showing the total sales by month and by childcare category.

```
SET @sql = Null;
SELECT GROUP_CONCAT(DISTINCT CONCAT('sum(',case when S.Limit_Mins = L.Limit_Mins and
S.Childcare_Flag=1 then case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then D.Discount_Price
else P.Retail_price end*Sa.Quantity else 0 end,') as ',case when S.Limit_Mins = L.Limit_Mins and
S.Childcare_Flag=1 then S.Limit_Mins else 'Extra' end)order by S.Limit_Mins asc)
INTO @sql
from `Limit` L
```

```
join Store S on
L.Limit_Mins = S.Limit_Mins
join Sale Sa on
S.Store_Number=Sa.Store_number
join `Date` Dt on
Dt.`Date`=Sa.`Date`
join Product P on
Sa.PID=P.PID
left outer join Discount D
on Sa.PID=D.PID
and Sa.`Date`=D.Discount_Date;
SET @sql = CONCAT('SELECT month(Sa.`Date`),',@sql,'sum(case when S.Childcare_Flag = 0 then  case when D.PID =Sa.PID and D.Discount_Date=Sa.`Date` then D.Discount_Price else P.Retail_price end*Sa.Quantity end) as `No Childcare`',' from
`Limit` L
join Store S on L.Limit_Mins = S.Limit_Mins
join Sale Sa on
S.Store_Number=Sa.Store_number
join `Date` Dt on
Dt.`Date`=Sa.`Date`
join Product P on
Sa.PID=P.PID
left outer join Discount D
on Sa.PID=D.PID
and Sa.`Date`=D.Discount_Date group by Sa.`Date`
where Sa.`Date`> now() - INTERVAL 12 month
group by 1');
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

Report 8 Restaurant Impact on Category Sales

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 8 Restaurant Impact on Category Sales*** button, then the system runs **Report 8** task:

- Get Store Names from Store with and without Restaurant and conditionally format it as Store Type.
- Combine on Product ID via Product to Category to get the Category Name.
- Combine with Sale to get count of Quantity of products sold by Category Name by Store Type.
- Filter out Category Names without Product Names in it.
- Display by category Name ascending and Store Type Ascending.

```
select C.Category_Name,
case when S.Restaurant_Flag=1 then 'Restaurant'
else 'Non-restaurant' end as `Store Type`,
sum(Sa.Quantity)
from Store S join Sale Sa
on S.Store_Number=Sa.Store_number
join Product P on
P.PID=Sa.PID
join Category C
on C.PID=P.PID
group by 1,2;
```

Report 9 Advertising Campaign Analysis

Abstract Code

User Enters the application/Dashboard and clicks on the ***Report 9 Advertising Campaign Analysis*** button, then the system runs **Report 9** task:

- Retrieve Product ID, Product Name from Product
- CombineDiscount to filter Product IDs that had a discount price.
- Combine with Store to get Store Number.
- Combine with Campaign to get the campaign dates.
- Combine with Date and filter on Sale to get count of products sold during and outside campaign dates.
- Calculate the difference between the two totals which have been calculated above.
- Sort the results by difference in descending.
- Display the top 10 followed by bottom 10 from the above result.

```
select A.`Product ID`, A.`Product Name`,sum(`Sold During Campaign`),sum(A.`Sold Outside Campaign`),sum(A.`Difference`) from (

(select P.PID as `Product ID`,P.Product_Name as `Product Name`,

sum(case when Ca.`date` is not null then Sa.Quantity else 0 end) as `Sold During Campaign`,

sum(case when Ca.`date` is null then Sa.Quantity else 0 end) as `Sold Outside Campaign`,

sum(case when Ca.`date` is not null then Sa.Quantity else 0 end )-sum(case when Ca.`date` is null then Sa.Quantity else 0 end) as `Difference`

from Sale Sa join Product P on P.PID=Sa.PID

left join Campaign Ca

on Ca.`Date`=Sa.`Date`

group by 1,2

order by `Difference` Desc

LIMIT 10)

union all(

select P.PID as `Product ID`,P.Product_Name as `Product Name`,

sum(case when Ca.`date` is not null then Sa.Quantity else 0 end) as `Sold During Campaign`,

sum(case when Ca.`date` is null then Sa.Quantity else 0 end) as `Sold Outside Campaign`,

sum(case when Ca.`date` is not null then Sa.Quantity else 0 end )-sum(case when Ca.`date` is null then Sa.Quantity else 0 end) as `Difference`
```

```
from Sale Sa join Product P on
P.PID=Sa.PID
left join Campaign Ca
on Ca.`Date`=Sa.`Date`
group by 1,2
order by `Difference` ASC
LIMIT 10)
)A group by 1,2;
```