Table of Contents:

Main Menu

Abstract Code

- Display sum count of Store, total Count of Product

```
SELECT count(Store_Number) FROM store s
```

- View the count of store who provide Restaurant or Snack_bar service

```
SELECT count(Store_Number) FROM store s
where s.Restaurant = true or s.Snack_bar = true
```

- View the total count of store who provide Childcare service

```
SELECT count(Store_Number) FROM store s
where  s.limit_time is NOT null
```

- View the count of products

```
SELECT count(PID) FROM product p
```

- View the count of distinct advertising campaigns

```
SELECT count(ADname ) FROM ad a
```

View Holidays

Abstract Code

● User clicked on ***View Holidays*** from **Main Menu**:
● Run the **View Holidays** task: query for information about FESTIVAL using the system from the HTTP Session/Cookie
    ○ For each Holiday of FESTIVAL:
        ■ Display FESTIVAL.Name, FESTIVAL.Date

```
SELECT * FROM festival
```

Add Holidays

Abstract Code

● User clicked on ***Add Holiday*** from from **Main Menu**:
● Run the **Add Holiday** task: add information about FESTIVAL using the system from the HTTP Session/Cookie:
    ● User selects date in a date drop down box
    ● User enters FESTIVAL.Name for the Holiday input field
        ● If data validation is successful for FESTIVAL.Name input field: Add Holiday into FESTIVAL

```
INSERT INTO festival(festival_date, fname) values ('$Date', '$Festival')
```

        ● Else: Go back to **Main Menu**, with error message

View Population of Cities

Abstract Code

● User clicked on ***View Population*** from **Main Menu**:
● Run the **View population** task: query for information about POPULATION using the system from the HTTP Session/Cookie For each city of CITY:
    ● Display CITY.State_name, CITY.City_name, CITY.Population

```
SELECT * FROM city
```

● Show ***Update Population*** button.

Update Population of Cities

Abstract Code

- User clicked on ***Update Population*** from **View Population**:
- Run the **Update Population** task: edit information about CITY using the system from the HTTP Session/Cookie:
    - User selects city in a city drop down box
    - User enters city name($city) and state name($state) for the City to be updated input field
    - User enters population($population) for the City to be updated input field
    - If data validation is successful for both City_name and Population input field:
        - Find current CITY using CITY.City_name == $city and CITY.State_name == $state
        - Edit CITY.Population = $population
    - Else:
        - Go back to **View Population** task, with error message

```
UPDATE city
SET population = '$population'
WHERE City_Name = '$city' and State_Name = '$state';
```

View Category Report

Abstract Code

- User click on ***View Category Report*** from **Main Menu**;
- Run the **View Category Report** task: query for information about CATEGORY using the system from the HTTP Session/Cookie;
- For each category of CATEGORY
    - Find CATEGORY.Name;
    - Find the total number of products belongs to this category;
    - Calculate the min price of products belongs to this category;
    - Calculate the average price of products belongs to this category;
    - Calculate the max price of products belongs to this category;
- Order by category name ascending.

```
SELECT main.cname, count(*) AS numbers, min(price), max(price), avg(price) FROM
(
   SELECT c.cname, a.pid FROM category c
   JOIN assigns a
   ON c.cname = a.cname
) main
JOIN product p
ON main.pid = p.pid
GROUP BY main.cname
ORDER BY main.cname
```

- Display all the above info in the **View Category Report.**

View Actual versus Predicted Revenue for Couches and Sofas Report

Abstract Code

● User click on ***Actual versus Predicted Revenue for Couches and Sofas*** from **Main Menu**;
● Run **View Actual versus Predicted Revenue for Couches and Sofas** task:query for information about CATEGORY using the system from the HTTP Session/Cookie;
- ● Find PRODUCT.Pname, PRODUCT.PID in PRODUCT;
- ● Combine PRODUCT and CATEGORY to limit the search scope as 'Couches and Sofas' (mark it as temporary t1), and find the PID, Pname, Price;

```
WITH t1 AS(
   SELECT p.pid, p.price, p.pname , a.cname FROM product p
   JOIN assigns a
   ON p.PID = a.PID
   WHERE cname= '$Couches and Sofas'
)
```

- ● Combine t1 and sales, we can get one new scope with sales information and limited category scope 'Couches and Sofas'; then we can find PID, Pname, Sell_quantity, Sell_date. We can rename all this info set as temporary t2.

```
SELECT t2.sell_date, t2.pid, t2.pname, t2.cname, t2.price, t2.sell_quantity
    ,(CASE WHEN percentage  IS NULL THEN 100 ELSE percentage END) AS
percentage
    ,(CASE WHEN percentage  IS NULL THEN 100 ELSE 75 END) AS
predict_precentage
FROM
(
   SELECT t1.pid, t1.price, t1.pname, t1.cname, s.Sell_quantity, s.Sell_date
FROM
    (
      SELECT p.pid, p.price, p.pname, a.cname FROM product p
      JOIN assigns a
      ON p.PID = a.PID
      WHERE cname= '$Couches and Sofas'
    ) t1
   JOIN sales s
   ON t1.pid = s.pid
   ) t2
LEFT JOIN discount dis
ON t2.pid = dis.pid AND t2.Sell_date = dis.discount_date
```

- Combine t2 and DISCOUNT with Sell_date then we can get all the sold goods' info (Sell_date, PID, Pname, Price, Percentage, DISCOUNT.Date, Sell_amount) which we can use them to calculate the actual revenue and predicted revenue; if we find one product has discount date, we will set this date and this PID 's Sell_amount * 0.75 when calculate the predicted revenue.

```
SELECT pid,price,pname,sell_quantity,cname,percentage, predict_precentage,sell_date
    ,price*percentage / 100 *Sell_quantity AS actual_revenue
    ,price*predict_precentage / 100 *Sell_quantity AS predicted_revenue
FROM
(
    SELECT t2.sell_date, t2.pid, t2.pname, t2.cname, t2.price, t2.sell_quantity
            ,(CASE WHEN percentage  IS NULL THEN 100 ELSE percentage END) AS
percentage
            ,(CASE WHEN percentage  IS NULL THEN 100 ELSE 75 END) AS
predict_precentage
    FROM
    (
        SELECT t1.pid, t1.price, t1.pname, t1.cname, s.sell_quantity, s.sell_date
        FROM
        (
            SELECT p.pid, p.price, p.pname, a.cname FROM product p
            JOIN assigns a
            ON p.pid = a.pid
            WHERE cname= '$Couches and SofAS'
        ) t1
        JOIN sales s
        ON t1.pid = s.pid
   ) t2
LEFT JOIN discount dis
ON t2.pid = dis.pid and t2.sell_date = dis.discount_date
) t3
```

- Calculate the gap between the actual revenue and predicted revenue, and list the gap > 5000 or < -5000 and order them by descending.

```
SELECT pid, pname, price,(sold_on_retail + sold_on_discount) AS
total_sale_quantity, sold_on_retail, sold_on_discount, gap FROM
(
      SELECT pid, pname, price, sum(CASE WHEN percentage = 100 THEN
sell_quantity ELSE 0 END) AS sold_on_retail,
        sum(CASE WHEN percentage != 100 THEN sell_quantity ELSE 0 END) AS
sold_on_discount
        ,sum(predicted_revenue) AS  predicted_revenue_all, sum(actual_revenue) AS
```

```
actual_revenue_all
         ,(sum(predicted_revenue)  - sum(actual_revenue)) AS gap
FROM
       (
         SELECT pid,price,pname,sell_quantity,cname,percentage,
predict_precentage,sell_date
               ,price*percentage / 100 *Sell_quantity AS actual_revenue
               ,price*predict_precentage / 100 *Sell_quantity AS predicted_revenue
       FROM
       (
             SELECT t2.Sell_Date,t2.pid,t2.pname,t2.cname, t2.price, t2.Sell_quantity
             ,(CASE WHEN percentage  IS NULL THEN 100 ELSE percentage END) AS
percentage
             ,(CASE WHEN percentage  IS NULL THEN 100 ELSE 75 END) AS
predict_precentage
             FROM
             (
            SELECT t1.pid, t1.price, t1.pname, t1.cname, s.Sell_quantity, s.Sell_date
FROM
              (
                SELECT p.pid, p.price, p.pname , a.cname FROM product p
                JOIN assigns a
                ON p.PID = a.PID
                WHERE cname= 'Couches and Sofas'
              ) t1
            JOIN sales s
            ON t1.pid = s.pid
             ) t2
             LEFT JOIN discount dis
             ON t2.pid = dis.pid and t2.sell_date = dis.discount_date
         ) t3
       ) t4
GROUP BY pid
) t5
WHERE gap > 5000 OR gap < -5000
ORDER BY gap DESC
```

● Display all the above info in the **View Actual versus Predicted Revenue for Couches and Sofas.**

View Store Revenue by Year by State Report

Abstract Code

● When User click ***View Store Revenue by Year by State*** button from **Main Menu**
● Run ***View Store Revenue by Year by State*** task: query for information about DATE using the system from the HTTP Session/Cookie;
    ● User selects stateName('$State_name') in a date drop down box;
    ● Find the Store_number, Address, City_name from STORE and CITY with stateName provided by user, and rename this result set as temporary t1;

```
SELECT  t1.Store_Number, t1.Address, t1.City_name, d.pid, d.sell_date, d.Sell_Quantity FROM
(
    SELECT s.Store_Number, s.Address, s.City_name, s.State_Name FROM store s
    INNER JOIN city c
    ON s.City_Name = c.City_Name and s.State_Name = c.State_Name
    WHERE c.State_Name = '$state_name' and c.City_Name = '$city_name'
) t1
JOIN sales d
ON t1.Store_Number = d.Store_Number
```

    ● Find the Price,Sell_amount and Sell_date from t1 and SALES by combine them as the temporary set t2; then combine product

```
SELECT  Store_Number, Address, City_Name, sell_date,sell_quantity,price,
(CASE WHEN percentage IS NULL THEN 100 ELSE percentage END) AS percentage
FROM
    (
        SELECT t2.Store_Number, t2.Address, t2.City_name, t2.sell_quantity,
t2.Sell_date,t2.pid, p.price
        FROM
        (
            SELECT  t1.Store_Number, t1.Address, t1.City_name, d.pid, d.sell_date,
d.Sell_Quantity
        FROM
        (
            SELECT s.Store_Number, s.Address, s.City_name, s.State_Name FROM store s
            INNER JOIN city c
            ON s.City_Name = c.City_Name and s.State_Name = c.State_Name
            WHERE c.State_Name = '$state_name' and c.City_Name = '$city_name'
        ) t1
        JOIN sales d
        ON t1.Store_Number = d.Store_Number
        ) t2
        JOIN product p
        ON t2.pid = p.pid
```

```
)t3
LEFT JOIN discount dis
ON t3.pid = dis.pid and t3.sell_date = dis.discount_date
```

- Based on t2, we combine result of t2 and DISCOUNT then we can identify the price and discount percentage which we can use them to calculate the revenue;
- Order them by year(use inner function to get year from date.) ascending and then order revenue by descending.

```
SELECT  Store_Number, Address, City_Name,sales_year, sum_total_revenue FROM
(
  SELECT  Store_Number, Address, City_Name,sales_year , sum(total_revenue) AS
sum_total_revenue FROM
  (
    SELECT Store_Number, Address, City_Name,year(sell_date) AS sales_year,sell_quantity,
price,percentage,
          (price * sell_quantity * percentage / 100) AS total_revenue FROM
    (
      SELECT  Store_Number, Address, City_Name, sell_date,sell_quantity,price,
          (CASE WHEN percentage IS NULL THEN 100 ELSE percentage END) AS percentage
      FROM
      (
          SELECT t2.Store_Number, t2.Address, t2.City_name, t2.sell_quantity,
t2.Sell_date,t2.pid, p.price
          FROM
          (
              SELECT  t1.Store_Number, t1.Address, t1.City_name, d.pid, d.sell_date,
d.Sell_Quantity FROM
              (
               SELECT s.Store_Number, s.Address, s.City_name, s.State_Name FROM store s
                  INNER JOIN city c
                  ON s.City_Name = c.City_Name and s.State_Name = c.State_Name
                 WHERE c.State_Name = '$state_name' and c.City_Name = '$city_name'
               ) t1
               JOIN sales d
               ON t1.Store_Number = d.Store_Number
          ) t2
          JOIN product p
          ON t2.pid = p.pid
      ) t3
      LEFT JOIN discount dis
      ON t3.pid = dis.pid and t3.sell_date = dis.discount_date
    ) t4
  ) t5
GROUP BY (sales_year + Store_Number)
) t6
ORDER BY sum_total_revenue DESC
```


● Display the year, revenue,Store_number, Address,City_name, in the ***View Store Revenue by Year by State.***

View Outdoor Furniture Sale on Groundhog Day Report

Abstract Code

● User clicked on ***Outdoor Furniture Sale on Groundhog Day*** button from **Main Menu**:
● Run the **View Outdoor Furniture Sale on Groundhog Day** task: query for information regarding sales of Outdoor Furniture category using the system from the HTTP Session/Cookie;
● FindPID in Outdoor Furniture category through ASSIGNS;

```
WITH c1 AS (SELECT pid FROM assigns WHERE cname = 'Outdoor Furniture')
```

● Combine with SALES on PID and calculate the sum, group by year;
select sum(amount), sum(amount) / 365 from sells where pid in category_pid group by year
● Find PID in Outdoor Furniture category through ASSIGNS;
    ● Combine with SALES on PID on 2nd, February, the Groundhog Day;
    Select sum(amount) from sells where pid in category_pid and month is 2, date is 2
    ● Divide Outdoor Furniture annual sale with 365 to get average daily sale;

```
SELECT year(s.sell_date) AS s_year
, sum(s.sell_quantity) AS 'total number of items sold in outdoor furniture category'
, sum(s.sell_quantity) / 365 AS 'average number of units sold per day'
, sum(CASE WHEN month(s.sell_date) = 2 AND day(s.sell_date) = 2 THEN s.sell_quantity ELSE 0 END) AS 'total number of units sold on groundhog day'
FROM sales s
JOIN c1
ON s.pid = c1.pid
GROUP BY s_year
```

● Order in ascending.

```
ORDER BY s_year ASC
```

● When ready, user selects the next action from choices in the **Main Menu**.

View State with Highest Volume for each Category Report

Abstract Code

● User clicked on ***State with Highest Volume for each Category*** from **Main Menu**:
● Run the **State with Highest Volume for each Category** task: query for information about category using the system from the HTTP Session/Cookie;
● User selects year ('$Year') and month ('$Month') in a date drop down box;
    ● Find entries matching year and month selected on SALES;

```
WITH t1 AS (select s.sell_date, s.sell_quantity, s.pid, st.state_name
            from (select *
                  from sales
                  where year(sell_date) = 2018 and month(sell_date) = 2) s
            join store st
            on s.store_number = st.store_number
),
```

    ● For each category matching in ASSIGNS, combine with STORE and CITY, find the products sold, sum the sales amount that group by state within the time frame;

```
t3 AS (SELECT cname AS category
  , state_name AS state
  , sum(sell_quantity) AS total_units_sold
  FROM (SELECT a.cname, t1.sell_date, t1.sell_quantity, t1.state_name
        FROM t1
        JOIN assigns a
        ON a.pid = t1.pid
        ) t2
  GROUP BY category, state
),
```

    ● List category name and list twice if there is a tie of 2 or more states, with separate state names;

```
t4 AS (SELECT category
  , max(total_units_sold) AS max_units
```

```
  FROM t3
  GROUP BY category
)

SELECT category, state, total_units_sold
FROM t3
RIGHT JOIN t4
ON t3.total_units_sold = t4.max_units AND t3.category = t4.category
```

- Order by category name in ascending.

```
ORDER BY category ASC
```

- When ready, user selects the next action from choices in the **Main Menu**.

View Revenue by Population Report

Abstract Code

- User click on ***View Revenue by Population Report*** from the **Main Menu.**
- Run the **View Revenue by stores and time task**:
  - For every SALES left join DISCOUNT to find the PID, Sell_date and Sell_amount and the Percentage if the SALES.Sell_date is equal to the DISCOUNT.Date and SALES.PID = DISCOUNT.PID. Rename this table as t1.

```
SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
, d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.pid = d.pid
AND s.sell_date  = d.discount_date
```

  - Replace the null value to 100, those days have no discount, rename this table as t2.

```
SELECT t1.PID
```

```
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100)
AS perc
FROM
(SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.PID = d.pid AND s.sell_date  = d.discount_date ) t1
```

- Table t2 left joins STORE on t2.store_number = STORE.store_number, to identify which city the store is located, rename the table as t3.

```
SELECT t2.PID
, t2.Store_Number
, t2.sell_date
, t2.Sell_Quantity
, t2.perc
, s2.City_Name
, s2.State_Name
FROM (
SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc FROM (
SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.PID = d.pid
AND s.sell_date  = d.discount_date
) t1
) t2
LEFT JOIN store s2
ON t2.store_number = s2.Store_Number
```

- Table t3 left joins CITY, to find the population in each city, rename the table as t4

```
SELECT t3.PID
, t3.Store_Number
, t3.sell_date
, t3.Sell_Quantity
, t3.perc
, t3.City_Name
, t3.State_Name
,c.Population
FROM (
SELECT t2.PID
, t2.Store_Number
, t2.sell_date
, t2.Sell_Quantity
, t2.perc
, s2.City_Name
, s2.State_Name
FROM (
SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc
FROM (
SELECT s.PID, s.Store_Number, s.sell_date, s.Sell_Quantity, d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.PID = d.pid
AND s.sell_date  = d.discount_date ) t1
) t2
LEFT JOIN store s2
ON t2.store_number = s2.Store_Number ) t3
LEFT JOIN city c
ON c.City_Name = t3.city_name AND c.State_Name = t3.state_name
```

- Table t4 left joins PRODUCT, to find the price for each PID. Rename table as t5

```
SELECT t4.PID
, t4.Store_Number
, t4.sell_date
, t4.Sell_Quantity
, t4.perc
```

```
, t4.City_Name
, t4.State_Name
,t4.Population
, p.price
FROM (
SELECT t3.PID
, t3.Store_Number
, t3.sell_date
, t3.Sell_Quantity
, t3.perc
, t3.City_Name
, t3.State_Name
,c.Population
FROM (
SELECT t2.PID
, t2.Store_Number
, t2.sell_date
, t2.Sell_Quantity
, t2.perc
, s2.City_Name
, s2.State_Name
FROM (
SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100)
AS perc
FROM (
SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.PID = d.pid AND s.sell_date  = d.discount_date ) t1
LEFT JOIN store s2
ON t2.store_number = s2.Store_Number ) t3
LEFT JOIN city c
ON c.City_Name = t3.city_name AND c.State_Name = t3.state_name) t4
LEFT JOIN product p
ON t4.PID = p.PID
```

- Identify each city by its population as a new column, city_size. Simplify the sell_date as only year.

```
SELECT YEAR(sell_date) AS year
,pid ,store_number
,sell_quantity * perc * price /100 AS rev_temp
,city_name, state_name
, (CASE WHEN population < 3700000 THEN 'Small'
                    WHEN population >=3700000 AND population<6700000 THEN
'Medium'
                    WHEN population >=6700000 AND population<9000000 THEN 'Large'
                    WHEN population >=9000000 THEN 'Extra Large' end)
AS city_size
FROM
(SELECT t4.PID
, t4.Store_Number
, t4.sell_date
, t4.Sell_Quantity
, t4.perc
, t4.City_Name
, t4.State_Name
,t4.Population
, p.price
FROM (
SELECT t3.PID
, t3.Store_Number
, t3.sell_date
, t3.Sell_Quantity
, t3.perc
, t3.City_Name
, t3.State_Name
,c.Population FROM (
 SELECT t2.PID
, t2.Store_Number
, t2.sell_date
, t2.Sell_Quantity
, t2.perc
, s2.City_Name
, s2.State_Name
FROM (
SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100)
AS perc
FROM (
SELECT s.PID
```

```
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
, d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.PID = d.pid
AND s.sell_date  = d.discount_date ) t1
) t2
LEFT JOIN Store s2
ON t2.store_number = s2.Store_Number )t3
LEFT JOIN city c
ON c.City_Name = t3.city_name AND c.State_Name = t3.state_name) t4
LEFT JOIN product p
ON t4.PID = p.PID
) t5
```

- Sum up all revenues by different city size and year with order of Sell_date FROM oldest to newest.

```
SELECT year
, city_size
, SUM(rev_temp) AS rev_by_yearlyCitySize
FROM
(SELECT Year(sell_date) AS year
,pid ,store_number
,sell_quantity * perc * price /100
AS rev_temp
,city_name
, state_name
, (CASE WHEN population < 3700000 THEN 'Small'
        WHEN population >=3700000 AND population<6700000 THEN 'Medium'
        WHEN population >=6700000 AND population<9000000 THEN 'Large'
        WHEN population >=9000000 THEN 'Extra Large' end )
AS city_size
FROM
(SELECT t4.PID
, t4.Store_Number
, t4.sell_date
, t4.Sell_Quantity
, t4.perc
, t4.City_Name
, t4.State_Name
, t4.Population
```

```
, p.price
FROM
(SELECT t3.PID
, t3.Store_Number
, t3.sell_date
, t3.Sell_Quantity
, t3.perc
, t3.City_Name
, t3.State_Name
,c.Population FROM
(
SELECT t2.PID
, t2.Store_Number
, t2.sell_date
, t2.Sell_Quantity
, t2.perc
, s2.City_Name
, s2.State_Name
FROM
(SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc FROM
(SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage FROM Discount d
RIGHT JOIN Sales s
ON s.PID = d.pid AND s.sell_date  = d.discount_date )t1
 ) t2
LEFT JOIN store s2
ON t2.store_number = s2.Store_Number )t3
LEFT JOIN city c
ON c.City_Name = t3.city_name AND c.State_Name = t3.state_name) t4
LEFT JOIN product p
ON t4.PID = p.PID
) t5 ) t6
GROUP BY year, city_size
ORDER BY YEAR(sell_date) ASC, city_size DESC
```

View Childcare Sales Volume Report

Abstract Code：

- User click on ***View Revenue by Childcare Report*** from **Main Menu.**
- SALES left joins PRODUCT to find the percentage happened in each sell_date. Rename the result as t2.

```
SELECT s.PID, s.Store_Number, s.sell_date, s.Sell_Quantity,d.percentage FROM
discount d
Right join sales s
on s.PID = d.pid and s.sell_date  = d.discount_date
```

- Replace the null value to 100, those days have no discount and find data in the last 12 month, rename the result as t3.

```
SELECT * FROM
(SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc FROM
(SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.PID = d.pid
AND s.sell_date  = d.discount_date ) t1
)t2
WHERE sell_date BETWEEN '$date1' AND '$date2'
```

- Table t3 left joins PRODUCT, to find the price for each PID, rename the result as t4.

```
SELECT t3.PID
, t3.store_number
, t3.sell_date
, t3.sell_quantity
, t3.perc, p.price FROM
(SELECT * FROM
(SELECT t1.PID
```

```
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc
FROM (
SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage
FROM discount d
RIGHT JOIN sales s
ON s.PID = d.pid AND s.sell_date  = d.discount_date )t1)t2
WHERE sell_date BETWEEN '$date1' AND '$date2') t3
LEFT JOIN product p
ON p.PID =t3.PID
```

- Table t4 left joins STORE to find the limit_time for each store. Stores which do not offer childcare will show null in limit_time. Rename table as t5.

```
SELECT t4.PID
, t4.store_number
, t4.sell_date
, t4.sell_quantity
, t4.perc
, t4.price
, s2.limit_time FROM
(SELECT t3.PID
, t3.store_number
, t3.sell_date
, t3.sell_quantity
, t3.perc
, p.price FROM
(SELECT * FROM
(SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc FROM
(SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage
```

```
FROM discount d
        RIGHT JOIN sales s
        ON s.PID = d.pid AND s.sell_date  = d.discount_date ) t1
)t2
        WHERE sell_date BETWEEN '$date1' AND '$date2')t3
        LEFT JOIN product p
        ON p.PID =t3.PID
)t4
        LEFT JOIN Store s2
        ON t4.store_number = s2.Store_Number
```

- In table t5 update the sell_date to year and month only and calculate, and calculate the revenue of each sell_date.

```
SELECT DATE_FORMAT
(sell_date,'%Y-%m') AS YM
,t5.sell_quantity * t5.perc * t5.price /100 AS temp_rev
, t5.limit_time
 FROM
(SELECT t4.PID
, t4.store_number
, t4.sell_date
, t4.sell_quantity
, t4.perc
, t4.price
, s2.limit_time
FROM
(SELECT t3.PID
, t3.store_number
, t3.sell_date
, t3.sell_quantity
, t3.perc, p.price
FROM
(SELECT * FROM
(SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc FROM (
        SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage FROM discount d
RIGHT JOIN sales s
```

```
ON s.PID = d.pid AND s.sell_date  = d.discount_date ) t1
) t2
WHERE sell_date BETWEEN '$date1' AND '$date2') t3
LEFT JOIN product p
        ON p.PID =t3.PID) t4
        LEFT JOIN store s2
        ON t4.store_number = s2.Store_Number
         ) t5
```

- Sum up the revenue with same YM and limit_time as rev_monthly_limit. Replace the null value in limi_time to “No childcare”

```
SELECT t6.YM
, SUM(temp_rev) AS rev_by_monthly_limit
, IFNULL(limit_time, 'No childcare') AS limit_time
FROM
(SELECT DATE_FORMAT(sell_date,'%Y-%m')AS YM
,t5.sell_quantity * t5.perc * t5.price /100 AS temp_rev
, t5.limit_time
FROM
(SELECT t4.PID
, t4.store_number
, t4.sell_date
, t4.sell_quantity
, t4.perc
, t4.price
, s2.limit_time
FROM
(SELECT t3.PID
, t3.store_number
, t3.sell_date
, t3.sell_quantity
, t3.perc, p.price
FROM
(SELECT * FROM
(SELECT t1.PID
, t1.Store_Number
, t1.sell_date
, t1.Sell_Quantity
, IFNULL(t1.percentage, 100) AS perc FROM
(SELECT s.PID
, s.Store_Number
, s.sell_date
, s.Sell_Quantity
,d.percentage FROM discount d
```

```
RIGHT JOIN sales s
ON s.PID = d.pid AND s.sell_date  = d.discount_date ) t1
)t2
WHERE sell_date BETWEEN '$date1' AND '$date2' )t3
LEFT JOIN product p
ON p.PID = t3.PID) t4
LEFT JOIN store s2
ON t4.store_number = s2.Store_Number  )t5)t6
GROUP BY YM, limit_time
ORDER BY YM
```

View Restaurant Impact on Category Sales Report

Abstract Code

- User click **View Restaurant Impact on Category Sales Report** button from Main Menu;
- Run **View Restaurant Impact on Category Sales Report** task:
  - Find CATEGORY.Name, STORE.Restaurant products quantity sold for each store

```
WITH t1 AS (
SELECT d.PID
, d.discount_date
, s.sell_date  AS sold_date
, a.ADdate  AS ad_date
, SUM(s.Sell_Quantity) AS total_sold
FROM discount d
INNER JOIN sales s
ON d.PID = s.PID AND d.discount_date = s.sell_date
LEFT JOIN AD_date a
ON a.ADdate = s.sell_date
GROUP BY d.PID, d.discount_date , s.sell_date, a.ADdate
)
```

- Find Restaurant records
- Rename non-restaurant store data as '$Non-restaurant'
- Calculate total quantity of products sold for both '$Restaurant' and '$Non-restaurant'
- Order the report by category name
- Order the report by Store_Type to list non-restaurant data first

```
SELECT CName AS Category
, CASE WHEN st.Restaurant = True THEN 'Restaurant'
ELSE 'Non-Restaurant' END AS Store_Type
, SUM(quantity) AS Quantity_Sold
FROM t1
LEFT JOIN store st
```

```
ON t1.Store_Number = st.Store_Number
GROUP BY Category, Store_Type
ORDER BY Category, Store_Type
```

View Advertising Campaign Analysis Report

Abstract Code

- User click **View Advertising Campaign Analysis Report** from main menu;
- Run **View Advertising Campaign Analysis Report** task:
  - Find all PRODUCT.PID, PRODUCT.PName of product and DATE.Sell_amount in two cases:
    - Product sold time in AD.DATE
    - Product sold time not in AD.DATE

```
WITH t1 AS (
SELECT s.PID
, s.sell_date  AS sold_date
, a.ADdate  AS ad_date
, SUM(s.Sell_Quantity) AS Toal_Sold
FROM sales s
LEFT JOIN ad_date a
ON a.ADdate = s.sell_date
WHERE s.PID in (select distinct PID from discount)
GROUP BY s.PID, s.sell_date, a.ADdate
)
```

  - Calculate total quantity of product sold for each case

```
t2 AS (
SELECT t1.PID
, SUM(CASE WHEN t1.ad_date IS NOT NULL THEN t1.Toal_Sold ELSE 0 END)  AS
Sold_During_Campaign
, SUM(CASE WHEN t1.ad_date IS NULL THEN t1.Toal_Sold ELSE 0 END)  AS
```

```
Sold_Outside_Campaign
FROM t1
GROUP BY t1.PID
)
```

- Calculate difference between two totals

```
output_res AS (
SELECT t2.PID
, p.PName
, Sold_During_Campaign
, Sold_Outside_Campaign
, Sold_During_Campaign -  Sold_Outside_Campaign AS Difference
FROM t2
LEFT JOIN product p
ON p.PID = t2.PID
)
```

- Sort the difference in descending order
- Select top 10 and bottom 10 records

```
top10 AS
(
SELECT * FROM output_res ORDER BY Difference DESC
LIMIT 10
)
,

bottom10 AS
(
SELECT * FROM output_res ORDER BY Difference ASC
LIMIT 10
)
```

- Display selected records in single report

```
SELECT * FROM top10
```

```
UNION
SELECT * FROM bottom10
```