Table of Contents

Abstract Code

User Interface

Abstract Code:

- Show ***"View Category Report"***,***"View Holiday"***, ***"Edit Holiday"***, ***"View City Population"***, ***"Edit City Population"***, ***"View Actual versus Predicted Revenue for Couches and Sofas Report"***, ***"View Store Revenue by Year by State"***, ***"View Outdoor Furniture on Groundhog Day"***, ***"View State with Highest Volume for each Category"***, ***"View Revenue by Population"***, ***"View Childcare Sales Volume"***, ***"View Restaurant Impact on Category Sales"***, ***"View Advertising Campaign Analysis"*** tabs.
- Upon:
    - Click ***View Category Report*** button- Jump to **View Category Report** task.
    - Click ***View Holiday*** button- Jump to **View Holiday** task.
    - Click ***Edit Holiday*** button- Jump to **Edit Holiday** task.
    - Click ***View City Population*** button- Jump to **View City Population** task.
    - Click ***Edit City Population*** button- Jump to **Edit City Population** task.
    - Click ***View Actual versus Predicted Revenue for Couches and Sofas*** button- Jump to **View Actual versus Predicted Revenue for Couches and Sofas** task.
    - Click ***View Store Revenue by Year by State*** button - Jump to **View Store Revenue by Year by State** task.
    - Click ***View Outdoor Furniture on Groundhog Day*** button- Jump to **View Outdoor Furniture on Groundhog Day** task.
    - Click ***View State with Highest Volume for each Category*** button- Jump to **View State with Highest Volume for each Category** task.
    - Click ***View State with Revenue by Population*** button- Jump to **View Revenue by Population** task.
    - Click ***View Childcare Sales Volume*** button- Jump to **View Childcare Sales Volume** task.
    - Click ***View Restaurant Impact on Category Sales*** button- Jump to **View Restaurant Impact on Category Sales** task.
    - Click ***View Advertising Campaign Analysis*** button- Jump to **View Advertising Campaign Analysis** task.

View Holiday

Abstract Code:

- User clicked on ***View Holiday*** button from **Main Menu**:
- Run the **View Holiday** task: query for display all holiday information in database now

```
SELECT *
FROM Holiday;
```

Edit Holiday

Abstract Code:

- User clicked on ***Edit Holiday*** button from **Main Menu**:
- Run the **Edit Holiday** task: query for insert holiday information with $date and $holiday are user defined date and holiday name

```
INSERT INTO Holiday (date, holiday)
VALUES ($date, $holiday);
```

- Run the **View Holiday** task: query displaying added holiday:

```
SELECT *
FROM Holiday
WHERE date = $date;
```

View Population

Abstract Code:

- User clicked on ***View Population*** button from **Main Menu**:
- Run the **View Population** task: query for display all cities and their population information in database now

```
SELECT city_name, population
FROM City;
```

Edit Population

Abstract Code:

- User clicked on ***Edit Population*** button from **Main Menu**:
- Run the **Edit Population** task: query for edit population number with $city and $population are user defined city name and population number.

```
UPDATE City
SET population = $population,
      city_size = CASE WHEN $population < 3700000 THEN 'Small'
                       WHEN $population <6700000 THEN 'Medium'
                       WHEN $population <9000000 THEN 'Large'
                        ELSE 'Extra Large' END

WHERE city_name=$city;
```

- Run the **View Population** task: query displaying updated city with it's population:

```
SELECT city_name, population
FROM City
WHERE city_name = $city;
```

View Category Report

Abstract Code:

- User clicked on ***View Category Report*** button from **Main Menu**:
- Run the **View Category Report** task: query for information about the category_category name, total number of products in that category, and the average regular price (not including discount days) of all the products in that category.
- Sort the results by category name ascending.

```
SELECT Sales.quanlity_sold AS quanlity_sold, Product_Category.category_name AS CategoryName
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.SaleID
INNER JOIN Product_Category ON Product_Category.PID = Sales_Product.PID
GROUP BY CategoryName
ORDER BY CategoryName ASC;
```

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

View Actual versus Predicted Revenue for Couches and Sofas

Abstract Code:

- User clicked on ***View Actual versus Predicted Revenue for Couches and Sofas*** button from **Main Menu**:
- Run **View Actual versus Predicted Revenue for Couches and Sofas** task: query for the information about each product in Couches and Sofas category.
    - Display PID, ProductName.

```
SELECT PID, product_name
FROM Product
INNER JOIN Product_Category ON Product_Category.PID = Product.PID
WHERE Product_Category.category_name = “Couches and Sofas”;
```

    - Find the current price using the PID; display DiscountPrice, RegularPrice.

```
SELECT Product.PID AS PID, DiscountPrice.discount_price AS DiscountPrice,
Product.regular_price AS RegularPrice
FROM Product
INNER JOIN DiscountPrice ON Product.PID = DiscountPrice.PID
INNER JOIN Product_Category ON Product_Category.PID = Product.PID
WHERE Category.category_name = “Couches and Sofas”;
```

    - Find the total number of units ever sold
        - Calculate the total number of units sold at a discount (DiscountPrice* QuantitySold)
        - Find the total number of units sold at regular price (RegularPrice* QuantitySold)

```
SELECT PID, (TotalNumberOfUnitsSoldwithDiscountPrice +
TotalNumberOfUnitsSoldwithRegularPrice) AS TotalNumberOfUnits
```

```
FROM(
(SELECT Sales_Product.PID, SUM(Sales.quanlity_sold) AS
TotalNumberOfUnitsSoldwithDiscountPrice
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.saleID
INNER JOIN DiscountPrice ON DiscountPrice.PID = Sales_Product.PID
INNER JOIN Product_Category ON Product_Category.PID =  Sales_Product.PID
WHERE Category.category_name = “Couches and Sofas”
GROUP BY Sales_Product.PID)  AS DiscountPriceSoldQuantity
JOIN
(SELECT Sales_Product.PID, SUM(Sales.quanlity_sold) AS
TotalNumberOfUnitsSoldwithRegularPrice
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.saleID
INNER JOIN Product ON Product.PID = Sales_Product.Product_PID
INNER JOIN Product_Category ON Product_Category.PID = Product.PID
WHERE Category.category_name = “Couches and Sofas”
GROUP BY Sales_Product.PID
) AS RegularPriceSoldQuantity
ON RegularPriceSoldQuantity.PID = DiscountPriceSoldQuantity.PID
)   temp
;
```

- Find the actual revenue collected from all the sales of the product, the predicted revenue of the product on sale (based on 75% volume selling at regular price), and the difference between the actual revenue and the predicted revenue.
  - Calculate the actual revenue collected from all the sales of the product (RegularPrice/DiscountPrice * QuantitySold)
  - Calculate the predicted revenue of the product on sale (based on 75% volume selling at retail price) (RegularPrice * QuantitySold * 75%)
  - Calculate the difference between the actual revenue and the predicted revenue

```
WITH regular_sold AS
(SELECT (Revenue1+Revenue2) AS Actual_Revenue
FROM(
(SELECT SUM(Sales.quanlity_sold*DiscountPrice.discount_price) AS Revenue1,
YEAR(Date) AS Year
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.saleID
INNER JOIN DiscountPrice ON DiscountPrice.PID = Sales_Product.Product_PID
INNER JOIN Product_Category ON Product_Category.PID =
Sales_Product.Product_PID
WHERE Category.category_name = “Couches and Sofas”) AS temp_table_A
JOIN
(SELECT SUM(Sales.quanlity_sold*Product.regular_price) AS Revenue2,YEAR(Date)
AS Year
FROM Sales
```

```
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.Sales_SaleID
INNER JOIN Product ON Product.PID = Sales_Product.Product_PID
INNER JOIN Product_Category ON Product_Category.PID = Product.PID
WHERE Category.category_name = “Couches and Sofas”) AS AS temp_table_B
ON AS temp_table_A.Year =AS temp_table_B.Year
)  temp
),
predicted_sold AS
(SELECT SUM(Sales.quanlity_sold*Product.regular_price*75%) AS
Predicted_Revenue
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.Sales_SaleID
INNER JOIN Product ON Product.PID = Sales_Product.Product_PID
INNER JOIN Product_Category ON Product_Category.PID =
Sales_Product.Product_PID
WHERE Category.category_name = “Couches and Sofas”)

SELECT (Actual_Revenue - Predicted_Revenue) AS diff
FROM regular_sold, predicted_sold;
```

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

View Store Revenue by Year by State

Abstract Code:

- User clicked on ***View Store Revenue by Year by State*** button from **Main Menu**:
- Run the **View Store Revenue by Year by State** task: Display all the states in the Drop-down box.
    - Upon selection of the specified state
    - Display store_ID, street_address, city_name

```
SELECT store_ID, street_address, city_name
FROM Store;
```

    - For each year:
        - Calculate the total revenue for all stores for each year in the selected state (revenue calculation must consider the products were sold at a discounted price).
        - Sort the total revenue by each year in ascending order and then by revenue in descending order.

```
SELECT (Revenue1+Revenue2) AS Revenue, Year
FROM
(
(SELECT SUM(Sales.quanlity_sold*DiscountPrice.Discount_Price) AS Revenue1,
YEAR(date) AS Year
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.saleID
```

```
INNER JOIN DiscountPrice ON DiscountPrice.PID = Sales_Product.PID
LEFT JOIN City ON City.city_name = Sales.city_name
WHERE City.state=$State AND YEAR(Sales.date) = $Year
GROUP BY YEAR(date)
) AS DiscountRevenue
JOIN
(SELECT SUM(Sales.quanlity_sold*Product.regular_price) AS Revenue2,YEAR(Date)
AS Year
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.Sales_saleID
INNER JOIN Product ON Product.PID = Sales_Product.PID
LEFT JOIN City ON City.city_name = Sales.city_name
WHERE City.state=$State AND YEAR(Sales.state) = $Year
GROUP BY YEAR(date)
) AS RegularRevenue
ON DiscountRevenue.Year = RegularRevenue.Year
) temp

ORDER BY YEAR ASC, Revenue DESC
;
```

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

View Outdoor Furniture on Groundhog Day

Abstract Code:

- User clicked on ***View Outdoor Furniture on Groundhog Day*** button from **Main Menu**:
- Run the **View Outdoor Furniture on Groundhog Day** task: query for information about the total number of outdoor furniture sold each year on Groundhog Day, average number of outdoor furniture sold per day each year, total number of outdoor furniture sold each year on Groundhog Day.
- Filter and display date (February 2) of each year.
    - Calculate the total number of units sold that year and daily average in the Outdoor Furniture category
    - Calculate the total number of units sold on Groundhog day that year in the Outdoor Furniture category
    - Sort the total number of units in the Outdoor Furniture category of each year in ascending order

```
WITH outdoor_furniture_sold AS
(SELECT quanlity_sold
FROM Sales
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.saleID
INNER JOIN Product_Category ON Product_Category.PID =  Sales_Product.PID
WHERE Product_Category.category_name = “Outdoor Furniture”),

groudhogday_furniture_sold AS
(SELECT Year(Date) AS Year, SUM(quanlity_sold) AS groundhog_sold
```

```
FROM outdoor_furniture_sold
WHERE Month(Date) = 2 AND Day(Date) = 2
GROUP BY Year),

total_avg_sold AS
(SELECT Year(Date) AS Year, SUM(quantity_sold) AS total_sold,
(SUM(quanlity_sold)/365) AS avg_sold
FROM outdoor_furniture_sold)
GROUP BY Year)

SELECT groudhogday_furniture_sold.Year,
groudhogday_furniture_sold.groundhog_sold, total_sold, avg_sold
FROM total_avg_sold
LEFT JOIN  groudhogday_furniture_sold ON groudhogday_furniture_sold.Year =
total_avg_sold.Year
ORDER BY Year ASC;
```

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

View State with Highest Volume for each Category

Abstract Code:

- User clicked on ***View State with Highest Volume for each Category*** button from **Main Menu**: query for information about the the states that sell greatest number of units in each category, all stores in that states at a selection month and year
- Filter and display the selected date

```
SELECT Year(Date) AND Month(Date)
FROM Date
```

- Run the **View State with Highest Volume for each Category task:**
    - Filter by each category
    - Calculate the total volume of units sold for that category of each state.
    - Select the state by finding the highest volume of units for that category of the state by states descending order
    - Select the stores in that state with the highest quantity sold in each category
    - User clicks ***drill-down detail*** button for category
    - Drop-down detail for each rows filtered by state, category, and date to discover which store in each city has the highest sales volume

```
WITH state_sale AS(
SELECT Product_Category.category_name as category_name,
SUM(Sales.quantity_sold) as state_total_sold, City.state
FROM Store
INNER JOIN City ON City.city_name = Store.city_name
```

```
INNER JOIN Sales ON Store.store_ID = Sales.store_ID
INNER JOIN Sales-Product ON Sales.saleID = Sales_Product.saleID
INNER JOIN Product_Category ON Sales-Product.PID = Product_Category.PID
WHERE Year(Date) = $Year AND Month(Date) = $Month
GROUP BY City.state, Product_Category.category_name),
store_sale AS(
SELECT Store.store_ID, Product_Category.category_name AS category_name,
SUM(Sales.quantity_sold) AS store_total_sold
FROM Store
INNER JOIN Sales ON Store.store_ID = Sales.store_ID
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.saleID
INNER JOIN Product_Category ON Sales_Product.PID = Product_Category.PID
WHERE Year(Date) = $Year AND Month(Date) = $Month
GROUP BY Store.store_ID, Product_Category.category_name)

SELECT category_name, state
FROM (state_sale
LEFT JOIN store_sale ON state_sale.category_name = store_sale.category_name
Where (category_name, state_total_sold) IN (SELECT category_name,
MAX(state_total_sold) FROM state_sale GROUP BY category_name)) temp
ORDER BY catogory_name ASC;
```

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

View Revenue by Population

Abstract Code:

- User clicked on ***View Revenue by Population*** button from **Main Menu**: query for information about the revenue corresponds to cities with different sizes.
- Run the **View Revenue by Population** task:
    - Extract year from Date.date
    - For **each** CitySize level (Small, Medium, Large, Extra Large):
        - Calculate annual revenue for every year
        - Sort in ascending order :
            - years (oldest to newest)
              CitySize (small, medium, large, extra large)

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

```
SELECT total_sold.year,
        SUM(CASE WHEN city_size = 'Small' THEN total_revenue ELSE 0 END) AS
Small,
        SUM(CASE WHEN city_size = 'Medium' THEN total_revenue ELSE 0 END)
AS Medium,
        SUM(CASE WHEN city_size = 'Large' THEN total_revenue ELSE 0 END) AS
```

```
Large,
            SUM(CASE WHEN city_size = 'Extra Large' THEN total_revenue ELSE 0
END) AS Extra_Large
FROM
(
SELECT  YEAR(Date.date) as year, IFNULL(reg_revenue.total_reg,0) +
IFNULL(dis_revenue.total_dis,0) AS total_revenue
FROM Date
LEFT JOIN (
SELECT YEAR(Sales.date) as year, City.city_size,
SUM(Sales.quantity_sold*Product.regular_price) AS total_reg
FROM Sales, Sales_Product, Product, Store,City
WHERE Sales.saleID = Sales_Product.saleID
            AND Sales_Product.PID = Product.PID
            AND Sales.storeID = Store.storeID
            AND Store.city_name = City.city_name
            AND (Sales_Product.PID, Sales.date) not in (SELECT PID, date FROM
DiscountPrice)
GROUP BY YEAR(Sales.date), City.city_size)  AS reg_revenue
ON YEAR(Date.date) = reg_revenue.year
LEFT JOIN
(SELECT YEAR(Sales.date) as year, City.city_size, SUM(Sales.quantity_sold*
DiscountPrice.discount_price) AS total_dis
FROM Sales, Sales_Product, Product, DiscountPice, Store,City
WHERE Sales.saleID = Sales_Product.saleID
            AND Sales_Product.PID = Product.PID
            AND Product.PID = DiscountPrice.PID
            AND Sales.date = DisountPrice.date
            AND Sales.storeID = Store.storeID
            AND Store.city_name = City.city_name
GROUP BY YEAR(Sales.date), City.city_size)  AS dis_revenue
ON YEAR(Sales.date) = dis_revenue.year AND reg_revenue.city_size =
dis_revenue.city_size)  AS total_sold
GROUP BY total_sold.year
ORDER BY total_sold.year;
```

View Childcare Sales Volume

Abstract Code:

- User clicked on ***Childcare Sales Volume*** button from **Main Menu**:
- Run the **View Childcare Sales Volume** task:
    - Filter SalesDate to the last 12 months
    - Extract Date.Month from Date.Date
    - for each Date.Month (as row) and ChildcareLimit (as column):
        - Sum total sales

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

```
SELECT *
FROM
(
SELECT month, childcare_limit, SUM(total_sold) as total_sales
FROM (
SELECT MONTH(Sales.date) as month, Store.childcare_limit,
SUM(Sales.quantity_sold*Product.regular_price) AS total_sold
FROM Sales, Sales_Product, Product, Store
WHERE Sales.saleID = Sales_Product.saleID
        AND Sales_Product.PID = Product.PID
        AND Sales.storeID = Store.storeID
        AND (Sales_Product.PID, Sales.date) not in (SELECT PID, date FROM
DiscountPrice)
        AND Sales.date BETWEEN NOW()-365 AND NOW()
GROUP BY MONTH(Sales.date), Store.childcare_limit
UNION  ALL
SELECT MONTH(Sales.date) as month, Store.childcare_limit,
SUM(Sales.quantity_sold*DiscountPrice.discount_price) AS total_sold
FROM Sales, Sales_Product, Product, Store, DiscountPrice
WHERE Sales.saleID = Sales_Product.saleID
        AND Sales_Product.PID = Product.PID
        AND Sales.storeID = Store.storeID
        AND Product.PID = DiscountPrice.PID
        AND Sales.date = DisountPrice.date
        AND Sales.date BETWEEN NOW()-365 AND NOW()
GROUP BY MONTH(Sales.date), Store.childcare_limit) AS tol_sales
GROUP BY month, childcare_limit
ORDER BY month, childcare_limit) AS sales_data

PIVOT (SUM(IFNULL(total_sales,0))  FOR childcare_limit in (SELECT DISTINCT
childcare_limit FROM Store) ) AS p_sales_data
ORDER BY month;
```

View Restaurant Impact on Category Sales

Abstract Code:

- User clicked on ***Restaurant Impact on Category Sales*** button from **Main Menu**:
- Run the **View Restaurant Impact on Category Sales** task: query for information about the product sold quantity in stores with or without onsite restaurant.
    - Calculate quantity sold in each store for each category
    - Sum total quantity sold in stores with restaurant for each category
    - Sum total quantity sold in stores with no restaurant for each category
    - Union two tables, order by CategoryName and Store_Type

```
WITH store_sale_cat AS(
```

```
SELECT Store.store_ID, Product_Category.category_name as category_name,
SUM(Sales.quantity_sold) as total_sold
FROM Store
INNER JOIN Sales ON Store.store_ID = Sales.store_ID
INNER JOIN Sales_Product ON Sales.saleID = Sales_Product.saleID
INNER JOIN Product_Category ON Sales_Product.PID =
Product_Category.category_name
GROUP BY Store.store_ID, Product_Category.category_name)

SELECT *
FROM (
SELECT category_name, 'Restaurant' as store_type, SUM(total_sold) as
quantity_sold
FROM store_sale_cat
WHERE store_ID  IN (SELECT store_ID FROM Store_StoreAffiliates WHERE
store_affiliates = ''Restaurant' )
GROUP BY category_name
UNION ALL
SELECT category_name, 'Non-Restaurant' as store_type, SUM(total_sold) as
quantity_sold
FROM store_sale_cat
WHERE store_ID  NOT IN (SELECT store_ID FROM Store_StoreAffiliates WHERE
store_affiliates = ''Restaurant'  )
GROUP BY category_name
) t
ORDER BY catogory_name, store_type;
```

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

View Advertising Campaign Analysis

Abstract Code:

- User clicked on ***Advertising Campaign Analysis*** button from **Main Menu**:
- Run the **View Advertising Campaign Analysis** task: query for information about the product sold quantity during ad or non ad time.
    - Filter only discount products, which are sold during Ad Campaign.
    - Calculate total item sold during campaign for discount product
    - Calculate total item sold during non-campaign period for discount product
    - Calculate sold-quantity difference for each discount product
    - Get top 10 and bottom 10 products of sold-quantity difference and return table order by difference in descending order

```
WITH sale_item AS
```

```
(SELECT  Product.PID, DiscountPrice.Date AS sale_date,
DiscountPrice.discount_price AS sale_price
FROM Product
INNER JOIN DiscountPrice ON DiscountPrice.PID = Product.PID
WHERE DiscountPrice.date in (SELECT date FROM AdCampaign WHERE
ad_campaign_description IS NOT NULL)
),
compaign_sold AS
(SELECT sale_items.PID, SUM(Sales.quantity_sold) as total_sale_quantity
FROM Sales,  Sales_Product, sale_items
WHERE Sales.saleID = Sales_Product.saleID
AND Sales_Product.ID = sale_items.PID
AND Sales.date = sale_items.sale_date
GROUP BY sale_items.PID),

regular_sold AS
(SELECT sale_items.PID, SUM(Sales.quantity_sold) as total_sale_quantity
FROM Sales,  Sales_Product, sale_items
WHERE Sales.saleID = Sales_Product.saleID
AND Sales_Product.PID = sale_items.PID
AND Sales.date != sale_items.sale_date
GROUP BY sale_items.PID),
Diff as
(SELECT compaign_sold.PID, total_sale_quantity, total_reg_quantity
(total_sale_quantity - total_reg_quantity) AS diff
FROM compaign_sold, regular_sold
WHERE compaign_sold.PID = regular_sold.PID)

SELECT *
FROM(
SELECT Diff.PID, product_name, total_sale_quantity, total_reg_quantity, difference
FROM Diff
JOIN Product ON Diff.PID = Product.PID
ORDER BY Diff.difference DESC  LIMIT 10
UNION
SELECT Diff.PID, product_name, total_sale_quantity, total_reg_quantity, difference
FROM Diff
JOIN Product ON Diff.PID = Product.PID
ORDER BY Diff.difference LIMIT 10
) t
ORDER BY difference DESC;
```

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