Contents

Tasks

View Main Menu Screen

Abstract Code

- Show statistics: count of stores,

```
SELECT count(store_number)
FROM Store;
```

- count of stores that offer food (have a restaurant, a snack bar, or both)

```
SELECT count(store_number)
FROM Store
WHERE has_restaurant = TRUE OR has_snack_bar = TRUE;
```

- count of stores offering childcare

```
SELECT count(store_number)
FROM Store
```

```
WHERE childcare_limit IS NOT NULL;
```

- count of products

```
SELECT count(pid)
FROM Product;
```

- count of distinct advertising campaigns;

```
SELECT count(description)
FROM AdvertisingCampaign;
```

- Show Buttons to reports and other functionalities;
- Upon:
    - Click ***Report 1 Category Report*** button - Jump to the **View Category Report** task;
    - Click ***Report 2 Actual versus Predicted Revenue for Couches and Sofas*** button - Jump to the **View Actual versus Predicted Revenue for Couches and Sofas** task;
    - Click ***Report 3 Store Revenue by Year by State*** button - Jump to the **View Store Revenue by Year by State** task;
    - Click ***Report 4 Outdoor Furniture on Groundhog Day*** button - Jump to the **View Outdoor Furniture on Groundhog Day** task;
    - Click ***Report 5 State with Highest Volume for each Category*** button - Jump to the **View State with Highest Volume for each Category By Year and Month** task;
    - Click ***Report 6 Revenue by Population*** button - Jump to the **View Revenue by Population** task;
    - Click ***Report 7 Childcare Sales Volume*** button - Jump to the **View Childcare Sales Volume** task;
    - Click ***Report 8 Restaurant Impact on Category Sales*** button - Jump to the **View Restaurant Impact on Category Sales** task;
    - Click ***Report 9 Advertising Campaign Analysis*** button - Jump to the **View Advertising Campaign Analysis** task.

View / Add Holiday Information

Abstract Code

- Click ***View/Add Holiday Information*** button - jump to **View Holiday Information** and **Add holiday Information** screen;

- Click **View Holiday Information**, the holiday information is displayed. Holiday name and the date will be included;

```
SELECT name, date_number
FROM Holiday;
```

- Click ***Back***, return to **View Holiday Information** and **Add holiday information** screen;
- Click ***Add holiday information*** button, a form is dispalyed. User is allowed to add holiday name ($HolidayName) and the date($DateNumber). After the new holiday information is added, click ***Save***. The new information is updated and **View Holiday Information** is displayed;

```
INSERT INTO Holiday(date_number, name)
VALUES ('$DateNumber', '$HolidayName');
```

- If the user clicks ***Back***, return to **Main Menu**.

Update Population of a City

Abstract Code

- User clicked on ***Update Population of a City*** button from **Main Menu**;
- User selects a city ($CityName, $State) to be updated, and the Population is supposed to be displayed;

```
SELECT city_name, state, population
FROM City
WHERE city_name='$CityName' AND state='$State';
```

- User enters *population* ($Population) in the input field for the selected city, and clicks *Enter* button;
- Run **Update Population of a City** Task:
    - If entered data is not a positive integer, the data is invalid, so display error message;
    - If data validation is successful, update the city information in the database;

```
UPDATE City
SET population='$Population'
WHERE city_name='$CityName' AND state='$State';
```

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

View Category Report

Abstract Code

- User clicked on ***View Category Report*** button from **Main Menu**;
- Run the **View Category Report** task:
    - Run the **Category** task: Find name of each category;
    - For each category, using category name:
        - Run the **Product** task: Find product information that corresponds to certain category;
        - According to product information of certain category, calculate the minimum, maximum, and average retail price of all the products in that category;
        - Sort the report by category name in ascending order;

```
SELECT C.name, COUNT(P.pid), MIN(P.retail_price), AVG(P.retail_price),
MAX(P.retail_price)
FROM Category AS C, Product AS P, Classification AS CL
WHERE P.pid = CL.pid AND C.name = CL.name
GROUP BY C.name
ORDER BY C.name ASC;
```

- When ready, the user selects next action from choices in **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 the **View Actual versus Predicted Revenue for Couches and Sofas** task:
    - Find the Product using “Couches and Sofas” category, then get the Product ID, the name of the Product, the retail price of the Product;
    - For each product of “Couches and Sofas”, find the Sale using Product ID;
    - For each Sale, find the Date corresponding to Sale;
    - For each Date and the Product, find the Discount coresponding to the Date and the Product;
    - For each product in the “Couches and Sofas” category:
        - Calculate the actual revenue based on the price(either discount or retail price) and quantity;

        - Calculate the predicted revenue based on the retail price and assumed quantity (based on 75% volume actual selling);
        - Calculate the difference between the actual revenue and the predicted revenue;
        - Display and Store the predicted revenue differences greater than $5000 (positive or negative);
    - Sort the report by predicted revenue difference in descending order;

```
SELECT P.pid, P.name, P.retail_price,
T2.total_sold,
T1.total_sold_at_a_discount,
(T2.total_sold - T1.total_sold_at_a_discount) AS total_sold_at_retail_price,
((T2.total_sold - T1.total_sold_at_a_discount * 0.25) * P.retail_price + T1.diff)
AS actual_revenue,
((T2.total_sold - T1.total_sold_at_a_discount * 0.25) * P.retail_price) AS
predicted_revenue,
T1.diff AS difference
FROM Product AS P,
(
    SELECT P.pid, SUM(S.quantity) AS total_sold_at_a_discount,
SUM(D.discount_price * S.quantity - P.retail_price * (S.quantity * 0.75)) AS diff
    FROM Product AS P, Sale AS S, Discount AS D, Classification AS C
    WHERE C.name = 'Couches And Sofas' AND C.pid = P.pid AND S.pid = P.pid AND
S.date_number = D.date_number AND D.pid = P.pid
    GROUP BY P.pid
) AS T1,
(
    SELECT P.pid, SUM(S.quantity) AS total_sold
    FROM Product AS P, Sale AS S, Classification AS C
    WHERE C.name = 'Couches And Sofas' AND C.pid = P.pid AND S.pid = P.pid
    GROUP BY P.pid
) AS T2
WHERE P.pid = T1.pid AND P.pid = T2.pid AND (T1.diff > 5000 OR T1.diff < -5000);
```

- When ready, the user selects next action from choices in **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**;
- User select *State* ($State) in the drop-down box;

```
SELECT DISTINCT state FROM City;
```

- Run the **Store Revenue by Year by State** task:
    - Find all the City using *State*;

- Find all Product;
- For each City, Find the Store using City;
- For each Store and Product, find Sale and Date using Store and Product;
- For each Sale and Date, find the Discount corresponding to the Sale and Date;
- For each store in the state:
  - Show the store ID, street address, city name, sales year;
  - Calculate total revenue;
- Sort the report first by year in ascending order and then revenue in descending order;

```
SELECT ST.store_number, ST.street_address, ST.city_name,
YEAR(S.date_number) AS years,
SUM(T1.revenue_at_discount + T2.revenue_at_retail) AS total_revenue
FROM Store AS ST, Product as P, Sale AS S, Discount AS D,
(
    SELECT SUM(D.discount_price * S.quantity) AS revenue_at_discount,
YEAR(D.date_number) AS years, ST.store_number
    FROM Store AS ST, Sale AS S, Discount AS D
    WHERE S.date_number = D.date_number AND S.store_number = ST.store_number AND
ST.state = '$State' AND S.pid = D.pid
    GROUP BY YEAR(D.date_number), ST.store_number
) AS T1,
(
    SELECT SUM(P.retail_price * S.quantity) AS revenue_at_retail,
YEAR(D.date_number) AS years, ST.store_number
    FROM Store AS ST, Product as P, Sale AS S, Discount AS D
    WHERE S.date_number <> D.date_number AND S.store_number = ST.store_number AND
ST.state = '$State' AND S.pid = P.pid AND D.pid = S.pid
    GROUP BY YEAR(D.date_number), ST.store_number
) AS T2
WHERE T1.store_number = T2.store_number AND T1.years = T2.years AND
ST.store_number = T1.store_number
GROUP BY YEAR(S.date_number), ST.store_number
ORDER BY YEAR(S.date_number) ASC, total_revenue DESC;
```

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

View Report Outdoor Furniture on Groundhog Day

Abstract Code

- User clicked on ***View Report Outdoor Furniture on Groundhog day*** button from **Main Menu**.
- Run the **View Outdoor Furniture on Groundhog Day** task:
  - Find the Product using "Outdoor Furniture" category;

- For each Product, find Sale Quantity and Date of each Prodcut;
- Using Date for each year:
  - Calculate the total number of items sold that year in the outdoor furniture category;
  - Calculate the average number of units sold per day that year(assume a year is exactly 365 days);
  - Calculte the total number of units sold on February 2 of that year;
- For each year, show if the total number of units sold on Groundhog Day (February 2) each year is significantly higher than the average number of units sold per day;
- Sort the report on the year in ascending order;

```
SELECT
T2.annual_outdoor_furniture_sold AS annual_outdoor_furniture_sold,
ROUND(T2.annual_outdoor_furniture_sold / 365) AS avg_outdoor_furniture_sold,
T1.groundhog_outdoor_furniture_sold AS groundhog_outdoor_furniture_sold
FROM
(
    SELECT YEAR(S.date_number) AS years, SUM(S.quantity) AS
groundhog_outdoor_furniture_sold
    FROM Sale AS S, Classification AS C
    WHERE S.pid = C.pid AND C.name = 'Outdoor Furniture' AND DAY(S.date_number) =
'2' AND MONTH(S.date_number) = '2'
    GROUP BY YEAR(S.date_number)
) AS T1,
(
    SELECT YEAR(S.date_number) AS years, SUM(S.quantity) AS
annual_outdoor_furniture_sold
    FROM Sale AS S, Classification AS C
    WHERE S.pid = C.pid AND C.name = 'Outdoor Furniture'
    GROUP BY YEAR(S.date_number)
) AS T2
WHERE T1.years = T2.years
ORDER BY T1.years ASC;
```

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

View State with Highest Volume for Each Category By Year and Month

Abstract Code

- User clicked on ***State with Highest Volume for Each Category By Year and Month*** button from **Main Menu**;
- User selected $Year and $Month from the drop-down menu;
- Run the **State with Highest Volume for Each Category By Year and Month** task:

    - Find all the Date;
    - Find all the category name;
    - For each Category, find Product using Category name;
    - Find all the City and corresponding state;
    - For each City, find all the Store;
    - For each state, calculate the number of units sold in all stores of this state for each Category;
    - For each Category, display the state that sold the highest number of units in that Category and the number of units that were sold in that state;
- Sort by category name in ascending order
    - Each category will only be listed once unless two or more states tied for selling the highest number of units in that category;

```
SELECT C.name, ST.state, MAX(T.total_sale) AS max_total_sale
FROM Classification AS C, Store AS ST, Sale AS S,
(
    SELECT C.name, ST.state, SUM(S.quantity) as total_sale
    FROM Classification AS C, Store AS ST, Sale AS S
    WHERE YEAR(S.date_number) = '$Year' AND MONTH(S.date_number) = '$Month' AND
S.pid = C.pid AND S.store_number = ST.store_number
    GROUP BY C.name, ST.state
) AS T
WHERE YEAR(S.date_number) = '$Year' AND MONTH(S.date_number) = '$Month' AND S.pid
= C.pid AND S.store_number = ST.store_number AND T.state = ST.state AND T.name =
C.name
GROUP BY C.name, ST.state
ORDER BY C.name ASC;
```

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

View Revenue by Population

Abstract Code

- User clicked on ***Revenue by Population*** button from **Main Menu**;
- Run the **Revenue by Population** task:
    - Find all the City and sort them by "population": 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);
    - Find all the Product;
    - For each year, find Date using year;
        - For each City:

        - Find Store using City;
        - For each Store and Product, Find Sale using Store and Product;
        - Find Discount using Product and Date;
        - Find the retail price for each sold product;
        - Calculate the total revenue in each year;
    - Sort the row using year in ascending order and sort the column using city size in ascending order;
    - Display the total revenue for each year grouped by city categories;
- When ready, the user selects next action from choices in **Main Menu**.

```
SELECT T.years, T.scale, SUM(T.revenue)
FROM
(
    SELECT R.years, C.scale, R.revenue, C.sn
    FROM
    (
        SELECT C.city_name, C.state, C.population,
            CASE
                WHEN C.population < 3700000
                THEN "Small"
                WHEN C.population < 6700000
                THEN "Medium"
                WHEN C.population < 9000000
                THEN "Large"
                ELSE "Extra Large"
            END AS scale,
            CASE
                WHEN C.population < 3700000
                THEN 0
                WHEN C.population < 6700000
                THEN 1
                WHEN C.population < 9000000
                THEN 2
                ELSE 3
            END AS sn
        FROM City AS C
    ) AS C,
    (
        SELECT YEAR(S.date_number) AS years, ST.city_name, ST.state,
        SUM(T1.revenue_at_discount + T2.revenue_at_retail) AS revenue
        FROM Sale AS S, Store AS ST,
        (
            SELECT SUM(D.discount_price * S.quantity) AS revenue_at_discount,
YEAR(S.date_number) AS years, ST.city_name, ST.state
            FROM Store AS ST, Sale AS S, Discount AS D
            WHERE S.date_number = D.date_number AND S.pid = D.pid AND
S.store_number = ST.store_number
            GROUP BY YEAR(S.date_number), ST.city_name, ST.state
        ) AS T1,
        (
```

```
            SELECT SUM(P.retail_price * S.quantity) AS revenue_at_retail,
YEAR(S.date_number) AS years, ST.city_name, ST.state
            FROM Store AS ST, Sale AS S, Product AS P, Discount AS D
            WHERE S.date_number <> D.date_number AND S.pid = D.pid AND
S.store_number = ST.store_number
            GROUP BY YEAR(S.date_number), ST.city_name, ST.state
        ) AS T2
        WHERE T1.city_name = T2.city_name AND T1.state = T2.state AND T1.years =
T2.years
        GROUP BY YEAR(S.date_number), ST.city_name, ST.state
    ) AS R
    WHERE C.city_name = R.city_name AND C.state = R.state
) AS T
GROUP BY T.years, T.scale, T.sn
ORDER BY T.years ASC, T.sn ASC;
```

View Childcare Sales Volume

Abstract Code

- User clicks ***View Childcare Sales Volume*** button from **Main Menu**;
- Run the **Childcare Sales Volume** task:
    - Find all the Date of the last 12 months;
    - Find all the Product;
    - Find all the Store;
    - For each Store:
        - Find Timelimit using Childcare;
        - Find Sale using Store and Product;
        - Find Discount using Date, Sale and Product;
        - Calculate the total amount of sale of each month under each Timelimit category;
- Display the table with total amount of sale under each Timelimit for each month;
- When ready, the user selects next action from choices in **Main Menu**.

```
SELECT MONTH(S.date_number) AS months, ST.childcare_limit,
SUM(T2.revenue + T1.revenue) AS sales_volumes
FROM Sale AS S, Store AS ST,
(
    SELECT MONTH(S.date_number) AS months, ST.childcare_limit,
    SUM(P.retail_price * S.quantity) AS revenue
    FROM Discount AS D, Product AS P, Store AS ST, Sale AS S
    WHERE S.date_number <> D.date_number AND S.pid = D.pid AND S.store_number =
ST.store_number
    AND (YEAR(S.date_number) = YEAR(now()) AND MONTH(S.date_number) <
MONTH(NOW())) OR (MONTH(S.date_number) >= MONTH(NOW()) AND YEAR(S.date_number) =
YEAR(NOW()) - 1)
    GROUP BY MONTH(S.date_number), ST.childcare_limit
```

```
) AS T1,
(
    SELECT MONTH(S.date_number) AS months, ST.childcare_limit,
    SUM(D.discount_price * S.quantity) AS revenue
    FROM Discount AS D, Store AS ST, Sale AS S
    WHERE S.date_number = D.date_number AND S.pid = D.pid AND S.store_number =
ST.store_number
    AND  (YEAR(S.date_number) = YEAR(now()) AND MONTH(S.date_number) <
MONTH(NOW())) OR (MONTH(S.date_number) >= MONTH(NOW()) AND YEAR(S.date_number) =
YEAR(NOW()) - 1)
    GROUP BY MONTH(S.date_number), ST.childcare_limit
) AS T2
WHERE T1.months = T2.months AND T1.childcare_limit = T2.childcare_limit
AND (YEAR(S.date_number) = YEAR(now()) AND MONTH(S.date_number) < MONTH(NOW()))
OR (MONTH(S.date_number) >= MONTH(NOW()) AND YEAR(S.date_number) = YEAR(NOW()) -
1) AND S.store_number = ST.store_number
GROUP BY YEAR(S.date_number), MONTH(S.date_number), ST.childcare_limit
ORDER BY YEAR(S.date_number) ASC, MONTH(S.date_number) ASC;
```

View Restaurant Impact on Category Sales

Abstract Code

- User clicked on ***View Restaurant Impact on Category Sales*** button from **Main Menu**;
- Run the **Restaurant Impact on Category Sales** task:
    - Find the all the category name (may exclude any categories that are not assigned products) and if the store has a restaurant;
    - For each category:
        - Find the Product corresponding to the category;
        - For each Store without a restaurant, Find the Sale using Store, Product;
        - For each Store with a restaurant, Find the Sale using Store, Product;
        - Calculate the total quantity of all products sold in the stores without a restaurant; Display the store type as “Restaurant” and the total Quantity Sold;
        - Calculate the total quantity of all products sold in the stores with a restaurant; Display the store type as “Non-restaurant” and the total Quantity Sold;
        - Calculate the total quantity of all products sold in the category;
- Sort the the results by category name in ascending order and with non-restaurant store data listed first;

```
SELECT C.name, Sum(S.quantity) As total_products, ST.has_restaurant
FROM Product As P, Classification AS C, Sale As S, Store As ST
```

```
WHERE P.pid = C.pid AND S.pid = P.pid And S.store_number= ST.store_number
GROUP BY C.name, ST.has_restaurant
ORDER BY C.name ASC, CASE WHEN ST.has_restaurant THEN 1 ELSE 0 END ASC;
```

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

View Advertising Campaign Analysis

Abstract Code

- User clicked on ***Advertising Campaign Analysis*** button from **Main Menu**;
- Run the **Advertising Campaign Analysis** task:
    - Find the Product ID and Product Name;
    - For each Product, find Discount using Product
    - Find Date using Discount;
    - For each Date, find Advertising Campaign dates using Date;
    - For each Product:
        - Find Sale Quantity;
        - Calculate total sale quantity when Advertising Campaign is active and when Advertising Campaign is not active;
        - Compute the difference between the two total sales;
    - Display the results for each Product:
        - Sort the the results by difference in descending order;
        - Only display the top 10 and the bottom 10 results;
- When ready, the user selects next action from choices in **Main Menu**.

```
SELECT T.pid, T.name, T.sale_during_campaign, T.sale_outside_campaign,
T.difference
FROM
(
    (
        SELECT P.pid, P.name, T1.sale_during_campaign, T2.sale_outside_campaign,
(T1.sale_during_campaign - T2.sale_outside_campaign) AS difference
        FROM Product AS P,
          (
              SELECT P.pid, SUM(S.quantity) AS sale_during_campaign
              FROM Product AS P, Sale AS S, Discount AS DS, Occur AS O
              WHERE DS.date_number = S.date_number AND DS.pid = S.pid AND
DS.date_number = O.date_number
              GROUP BY P.pid
          ) AS T1,
          (
              SELECT P.pid, SUM(S.quantity) AS sale_outside_campaign
              FROM Product AS P, Sale AS S, Discount AS DS, Occur AS O
              WHERE DS.date_number = S.date_number AND DS.pid = S.pid AND
DS.date_number <> O.date_number
```

```
            GROUP BY P.pid
          ) AS T2
        WHERE T1.pid = T2.pid AND T1.pid = P.pid AND T2.pid= P.pid
        GROUP BY P.pid
        ORDER BY difference DESC
        LIMIT 10
    )
    UNION
    (
        SELECT P.pid, P.name, T1.sale_during_campaign, T2.sale_outside_campaign,
(T1.sale_during_campaign - T2.sale_outside_campaign) AS difference
        FROM Product AS P,
          (
            SELECT P.pid, SUM(S.quantity) AS sale_during_campaign
            FROM Product AS P, Sale AS S, Discount AS DS, Occur AS O
            WHERE DS.date_number = S.date_number AND DS.pid = S.pid AND
DS.date_number = O.date_number
            GROUP BY P.pid
          ) AS T1,
          (
            SELECT P.pid, SUM(S.quantity) AS sale_outside_campaign
            FROM Product AS P, Sale AS S, Discount AS DS, Occur AS O
            WHERE DS.date_number = S.date_number AND DS.pid = S.pid AND
DS.date_number <> O.date_number
            GROUP BY P.pid
          ) AS T2
        WHERE T1.pid = T2.pid AND T1.pid = P.pid AND T2.pid= P.pid
        GROUP BY P.pid
        ORDER BY difference ASC
        LIMIT 10
    )
) AS T
ORDER BY T.difference DESC;
```