Table of Contents:

Data Types

Store

Attribute	Data type	Nullable
storeID	Integer	Not Null
childcare_limit	Integer	Not Null
has_restaurant	Boolean	Not Null
has_snack_bar	Boolean	Not Null
phone_number	String	Not Null
street_address	String	Not Null

City

Attribute	Data type	Nullable
city_name	String	Not Null
state	String	Not Null
population	Integer	Not Null

Sold

Attribute	Data type	Nullable
PID	Integer	Not Null
storeID	Integer	Not Null
date_attr	Date	Not Null
quantity	Integer	Not Null

Product

Attribute	Data type	Nullable
PID	Integer	Not Null
product_name	String	Not Null
retail_price	Float	Not Null

DateYMD

Attribute	**Data type**	**Nullable**
date_attr	Date	Not Null

BelongTo

Attribute	**Data type**	**Nullable**
PID	Integer	Not Null
category_name	String	Not Null

Category

Attribute	**Data type**	**Nullable**
category_name	String	Not Null

Discount

Attribute	**Data type**	**Nullable**
PID	Integer	Not Null
date_attr	Date	Not Null
discount_price	Float	Not Null

Holiday

Attribute	**Data type**	**Nullable**
holiday_name	String	Not Null
date_attr	Date	Not Null

HasAdCamp

Attribute	**Data type**	**Nullable**
camp_description	String	Not Null
date_attr	Date	Not Null

AdCamp

Attribute	**Data type**	**Nullable**
camp_description	String	Not Null

Business Logic Constraints:

- Discount prices should not be higher than retail prices

Main Menu

Abstract Code

- Show "***Edit Holiday Information***", ***"Edit City Population***", ***"View Products by Category***", ***"Analyze Revenue for Couches and Sofas***", ***"View Store Revenue by Year by State***", ***"Analyze Sales for Outdoor Furniture on Groundhog Day***", ***"View States with Highest Volume Sold by Category***", ***"Calculate Revenue by Population Categories***", ***"Analyze Sales by Volume for each Childcare Category***", ***"Analyze Restaurant Impact on Sales by Categories***", and "***Analyze Sales during Advertising Campaigns***" buttons.

- When Main Menu is rendered, display the following statistics:
  the count of stores, stores offer food (restaurant and-or snack bar), stores offering childcare, products, distinct advertising campaigns.

```
SELECT COUNT(storeID) AS 'num_store'
FROM Store;

SELECT COUNT(storeID) AS 'num_storefood'
FROM Store
WHERE has_restaurant = 1 OR has_snack_bar = 1;

SELECT COUNT(storeID) AS 'num_childcare'
FROM Store
WHERE childcare_limit != 0;

SELECT COUNT(PID) AS 'num_product'
FROM Product;

SELECT COUNT(camp_description) AS 'num_camp'
FROM AdCamp;
```

- Upon:
  - Click ***Edit Holiday Information*** button: Jump to the **Holiday Information Form** task.
  - Click ***Edit City Population*** button: Jump to the **City Population Form** task.
  - Click ***View Products by Category*** button: Jump to the **Report 1** task.
  - Click ***Analyze Revenue for Couches and Sofas*** button: Jump to the **Report 2** task.
  - Click ***View Store Revenue by Year by State*** button: Jump to the **Report 3** task.
  - Click ***Analyze Sales for Outdoor Furniture on Groundhog Day*** button: Jump to the **Report 4** task.
  - Click ***View States with Highest Volume Sold by Category*** button: Jump to the **Report 5** task.

- Click ***Calculate Revenue by Population Categories*** button: Jump to the **Report 6** task.
- Click ***Analyze Sales by Volume for each Childcare Category*** button: Jump to the **Report 7** task.
- Click ***Analyze Restaurant Impact on Sales by Categories*** button: Jump to the **Report 8** task.
- Click ***Analyze Sales during Advertising Campaigns*** button: Jump to the **Report 9** task.

Holiday Information Form

Abstract Code

- called from ***Edit Holiday Information*** button from **Main Menu**:
- Upon:
    - Click ***Display Holiday*** button: Query database for current holidays (date and name) and display them.

```
SELECT holiday_name, date_attr
FROM Holiday
ORDER BY date_attr DESC;
```

    - User will fill in *holiday name* input fields as the value passed to 'holiday_name'.
    - User will choose *year, month, date* from provided fields in a drop list as the input for the 'holiday_date'
    - Click ***Add Holiday*** button:
        - If 'holiday_name' already exists, display warning message "this holiday name already exists, please change the name.'
        - else if 'holiday_name' or 'holiday_date' are null, display warning message "Holiday name or date cannot be null, please provide values"
        - else: insert new Holiday instance with those values.

```
INSERT INTO Holiday(holiday_name, date_attr)
VALUES(holiday_name, holiday_date)
```

    - Click ***Main Menu*** button: Jump back to the **Main Menu** task.
    - Click ***Close*** button: close this window.

City Population Form

Abstract Code

- called from ***Edit City Population*** button from **Main Menu**:
- User select the *city name* and *city state* that they wish to update from a drop list, passed to variables 'citychoose' and 'statechoose'.

- Upon:
    - Click ***Select City*** button: Query database for the *city name* and *city state* and display them.

```
SELECT *
FROM City
WHERE city_name = citychoose AND state = statechoose
```

    - User will fill in *city population* (newpop) input fields.
    - Click ***Update Population*** button:
        - If 'newpop' is a non-zero integer value, update City.population attribute with 'newpop'

```
UPDATE City
SET
    population = newpop
WHERE
    city_name = citychoose AND state = statechoose;
```

        - then execute the `SELECT` statement again to verify the change:

```
SELECT *
FROM City
WHERE city_name = citychoose AND state = statechoose
```

        - Otherwise, display warning message ('Population should be a non-zero integer number, please double-check.').

    - Click ***Main Menu*** button: Jump to the **Main Menu** task.
    - Click ***Close*** button: close this window.

Category Report

Abstract code

- User clicked on the ***Category Report*** button from the **Main Menu**.
- Run the **Category Report** task.
- User clicked on ***Report*** button, display the "Category Name", "Number of Products", "Minimum Price", "Average Price", and "Maximum Price" by executing the following SQL.

```
SELECT Category.category_name AS `Category Name`, COUNT(*) AS `Number of
Products`, IFNULL(MIN(Product.retail_price), 0) AS `Minimum Price`,
IFNULL(AVG(Product.retail_price), 0) AS `Average Price`,
IFNULL(MAX(Product.retail_price), 0) AS `Maximum Price`
FROM Category LEFT JOIN BelongTo ON Category.category_name =
BelongTo.category_name
LEFT JOIN Product ON BelongTo.PID = Product.PID
GROUP BY `Category Name`,
ORDER BY `Category Name` ASC;
```

- Upon
    - User click ***Main Menu*** button: return to **Main Menu**.

Actual versus Predicted Revenue for Couches and Sofas

Abstract code:

- User clicked on the ***Actual versus Predicted Revenue for Couches and Sofas*** button from the **Main Menu**.
- Run the **Actual versus Predicted Revenue for Couches and Sofas** task.

```
SELECT Product.product_name, ABS(`Predicted Revenue` - `Actual Revenue`) AS `Difference`
FROM
(SELECT Product.PID, SUM(Sold.quantity) AS `Total Units Sold`, 0.75 * `Total Units Sold` *
Product.retail_price AS `Predicted Revenue` FROM
(SELECT Product.PID, Product.product_name, Product.retail_price
FROM
BelongTo WHERE category_name = "Couches and Sofas"
LEFT JOIN Product ON BelongTo.PID = Product.PID
LEFT JOIN Sold ON Sold.PID = Product.PID
GROUP BY Product.product_name)
JOIN
SELECT Product.PID, SUM(Sold.quantity) AS `Total Retail Units Sold`
FROM
(SELECT Product.PID, Product.product_name, Product.retail_price
FROM
BelongTo WHERE category_name = "Couches and Sofas"
LEFT JOIN Product ON BelongTo.PID = Product.PID
LEFT JOIN Sold ON Sold.PID = Product.PID
GROUP BY Sold.PID) WHERE Sold.date_attr NOT IN (SELECT date_attr FROM DISCOUNT WHERE
PID = Sold.PID)
JOIN
SELECT Product.PID, SUM(Sold.quantity) AS `Total Discount Units Sold`
FROM
(SELECT Product.PID, Product.product_name, Product.retail_price
FROM
BelongTo WHERE category_name = "Couches and Sofas"
LEFT JOIN Product ON BelongTo.PID = Product.PID
LEFT JOIN Sold ON Sold.PID = Product.PID
GROUP BY Sold.PID) WHERE Sold.date_attr IN (SELECT date_attr FROM DISCOUNT WHERE PID
= Sold.PID))
WHERE `DIFFERENCE` > 5000
```

- Upon
    - User click ***Main Menu*** button: return to the **Main Menu**.

Store Revenue by Year by State

Abstract Code

- User clicked on ***Store Revenue by Year by State*** button from **Main Menu**:
- Run the **Store Revenue by Year by State** task.
- Find all available states from **City**.

```
SELECT DISTINCT state FROM City ORDER BY state;
```

- User select **$state** from a drop-down box.
- User clicked on ***Report*** button, display the "Store ID", "Store address", "City name", "Sales year", and "Total revenue" for the state by executing the following SQL

```
SET sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));
SELECT Store.storeID AS `Store ID`, Store.street_address AS `Store address`,
City.city_name AS `City name`, YEAR(Sold.date_attr) AS `Sales year`, SUM(IF(Sold.PID =
Discount.PID AND Sold.date_attr = Discount.date_attr, Discount.discount_price,
Product.retail_price) * Sold.quantity) AS `Total revenue`
FROM Sold
LEFT OUTER JOIN Discount ON Sold.PID = Discount.PID AND Sold.date_attr =
Discount.date_attr, Store, City, Product
WHERE Sold.storeID = Store.storeID AND Store.city_name = City.city_name AND
City.state = '$state ' AND  Sold.PID = Product.PID
GROUP BY Store.storeID, `Sales year`
ORDER BY `Sales year`, `Total revenue` DESC;
```

- click ***Main Menu*** button, return to **Main Menu**.
  User selects next action from choices in **Main Menu**.

Outdoor Furniture on Groundhog Day

Abstract Code

- User clicked on ***Outdoor Furniture on Groundhog Day*** button from **Main Menu**:
- Run the **Outdoor Furniture on Groundhog Day** task.
- User clicked on ***Report*** button, display the "Sales year", "Total units sold per year", "Avg units sold per day", and "Total units sold on GroundHog Day" by executing the following SQL

```
SET sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));
SELECT YEAR(Sold.date_attr) AS `Sales year`, SUM(Sold.quantity) AS `Total units sold
per year`, ROUND(SUM(Sold.quantity)/365, 0) AS `Avg units sold per day`,
Groundhog.total_quantity_sold_on_groundhog AS `Total units sold on GroundHog Day`
FROM Sold, BelongTo,
(SELECT SUM(Sold.quantity) AS total_quantity_sold_on_groundhog,
YEAR(Sold.date_attr) AS sold_year
FROM Sold, BelongTo
WHERE Sold.PID = BelongTo.PID AND BelongTo.category_name = 'Outdoor furniture'
AND DATE_FORMAT(Sold.date_attr,'%m-%d') = '02-02'
GROUP BY sold_year
) AS Groundhog
WHERE Sold.PID = BelongTo.PID AND BelongTo.category_name = 'Outdoor furniture'
AND Groundhog.sold_year = YEAR(Sold.date_attr)
GROUP BY `Sales year`
ORDER BY `Sales year`;
```

- click ***Main Menu*** button, return to **Main Menu**.
  User selects next action from choices in **Main Menu**.

View State with Highest Volume Sold by Category

Abstract Code

- User clicked on ***View States with highest volume sold by Category*** link from **Main Menu**
- Display **State with highest volume sold by Category** screen with the month/year dropdown list. The dropdown list is populated from:

```
SELECT
     DISTINCT DATE_FORMAT(Sold.date_attr, '%M-%Y')
FROM Sold
ORDER BY
     DATE_FORMAT(Sold.date_attr, '%M-%Y') ASC;
```

- Users choose month and year from available Date in the dropdown list.
- **Upon:**
    - ***View Report*** button click:
        - Clear previous display
        - Read the user's selected month/year from the dropdown list in the **State with highest volume sold by Category** screen
        - Display results of the following SQL:

```
SELECT Category, State, Units_Sold
FROM
      (SELECT unitsCategory AS Category, MAX(Units_Sold) AS Units_Sold
       FROM
            (SELECT Category.category_name AS unitsCategory, City.state,
SUM(quantity) AS Units_Sold
            FROM Sold
            JOIN Product ON Product.PID = Sold.PID
            JOIN Store ON Sold.storeID = Store.storeID
            JOIN City ON City.city_name = Store.city_name AND City.state =
Store.state
            JOIN BelongTo ON BelongTo.PID = Sold.PID
            JOIN Category ON Category.category_name =
BelongTo.category_name
           WHERE
                 MONTH(Sold.date_attr) = '$month' AND YEAR(Sold.date_attr)
= '$year'
            GROUP BY Category.category_name, City.state) AS Query1
      GROUP BY unitsCategory) AS Query2
          JOIN
           (SELECT Category.category_name AS stateCategory, City.state AS
State, SUM(quantity) AS stateUnits
                       FROM Sold
                       JOIN Product ON Product.PID = Sold.PID
            JOIN Store ON Sold.storeID = Store.storeID
            JOIN City ON City.city_name = Store.city_name AND City.state =
Store.state
            JOIN BelongTo ON BelongTo.PID = Sold.PID
            JOIN Category ON Category.category_name =
BelongTo.category_name
       WHERE
            MONTH(Sold.date_attr) = '$month' AND YEAR(Sold.date_attr) =
'$year'
        GROUP BY Category.category_name, City.state) AS Query3 ON Category =
stateCategory AND Units_Sold = stateUnits
ORDER BY Category ASC;
```

- User click ***Main Menu*** button: return to **Main Menu**.

Calculate Revenue by Population Categories

Abstract Code

- User clicks on ***Calculate revenue by population categories*** from **Main Menu**
- Display results of the following SQL:

```
SELECT
        YEARS.`Year`,
          IFNULL(smallCategory/smallCount, 0) AS `Small City`,
        IFNULL(mediumCategory/mediumCount, 0) AS `Medium City`,
        IFNULL(largeCategory/largeCount, 0) AS `Large City`,
        IFNULL(extraLargeCategory/extraLargeCount, 0) AS `Extra Large City`
FROM
        (SELECT DISTINCT YEAR(Sold.date_attr) AS `Year` FROM Sold) AS YEARS
          LEFT JOIN
                (SELECT
                        YEAR(Sold.date_attr) AS `Year`,
                            SUM(quantity * IF(ISNULL(Discount.discount_price), Product.retail_price,
Discount.discount_price)) AS `smallCategory`,
                        COUNT(City.city_name) AS `smallCount`
                FROM Sold
                JOIN Product ON Product.PID = Sold.PID
                JOIN Store ON Store.storeID = Sold.storeID
                JOIN City ON City.city_name = Store.city_name AND City.state = Store.state AND City.population < 3700000
                LEFT OUTER JOIN Discount ON Sold.date_attr = Discount.date_attr AND Sold.PID = Discount.PID
                GROUP BY YEAR(Sold.date_attr)) AS SMALL ON YEARS.`Year` = SMALL.`Year`
        LEFT JOIN
                (SELECT
                        YEAR(Sold.date_attr) AS `Year`,
                   SUM(quantity * IF(ISNULL(Discount.discount_price), Product.retail_price, Discount.discount_price)) AS
`mediumCategory`,
                        COUNT(City.city_name) AS `mediumCount`
                FROM Sold
                JOIN Product ON Product.PID = Sold.PID
                JOIN Store ON Store.storeID = Sold.storeID
                JOIN City ON City.city_name = Store.city_name AND City.state = Store.state AND City.population >= 3700000 AND
City.population < 6700000
                LEFT OUTER JOIN Discount ON Sold.date_attr = Discount.date_attr AND Sold.PID = Discount.PID
                GROUP BY YEAR(Sold.date_attr)) AS MEDIUM ON YEARS.`Year` = MEDIUM.`Year`
        LEFT JOIN
                (SELECT
                        YEAR(Sold.date_attr) AS `Year`,
                   SUM(quantity * IF(ISNULL(Discount.discount_price), Product.retail_price, Discount.discount_price)) AS
`largeCategory`,
                        COUNT(City.city_name) AS `largeCount`
                FROM Sold
                JOIN Product ON Product.PID = Sold.PID
                JOIN Store ON Store.storeID = Sold.storeID
                JOIN City ON City.city_name = Store.city_name AND City.state = Store.state AND City.population >= 6700000 AND
City.population < 9000000
                LEFT OUTER JOIN Discount ON Sold.date_attr = Discount.date_attr AND Sold.PID = Discount.PID
                GROUP BY YEAR(Sold.date_attr)) AS LARGE ON YEARS.`Year` = LARGE.`Year`
        LEFT JOIN
                (SELECT
                        YEAR(Sold.date_attr) AS `Year`,
                        SUM(quantity * IF(ISNULL(Discount.discount_price), Product.retail_price, Discount.discount_price)) AS
`extraLargeCategory`,
                        COUNT(City.city_name) AS `extralargeCount`
                FROM Sold
                JOIN Product ON Product.PID = Sold.PID
                JOIN Store ON Store.storeID = Sold.storeID
                JOIN City ON City.city_name = Store.city_name AND City.state = Store.state AND City.population >= 9000000
                LEFT OUTER JOIN Discount ON Sold.date_attr = Discount.date_attr AND Sold.PID = Discount.PID
                GROUP BY YEAR(Sold.date_attr)) AS EXTRALARGE ON YEARS.`Year` = EXTRALARGE.`Year`
ORDER BY Years.`Year`;
```

Childcare Sales Volume

Abstract Code

- User clicked on ***View Analyze Sales by volume for each Childcare category*** button from **Main Menu**:
- Run the ***Analyze Sales by volume for each Childcare category*** task.
- Display results of the following SQL:

```
SELECT
    MONTHS.`Month`,
    IFNULL(lessCategory / smallCount, 0) AS `Short
Childcare Service`,
    IFNULL(mediumCategory / mediumCount, 0) AS `Long
Childcare Service`,
    IFNULL(largeCategory / largeCount, 0) AS `Full
Childcare Service`,
    IFNULL(noneCategory / noneCount, 0) AS `No
Childcare Service`
FROM
    (SELECT DISTINCT
        MONTH(Sold.date_attr) AS `MONTH`
    FROM
        Sold) AS MONTHS
        LEFT JOIN
    (SELECT
        MONTH(Sold.date_attr) AS `MONTH`,
            SUM(quantity *
IF(ISNULL(Discount.discount_price),
Product.retail_price, Discount.discount_price)) AS
`lessCategory`,
            COUNT(Store.childcare_limit) AS
`smallCount`
```

```
 FROM
        Sold
    JOIN Product ON Product.PID = Sold.PID
    JOIN Store ON Store.childcare_limit
        AND Store.childcare_limit < 10
    LEFT OUTER JOIN Discount ON Sold.date_attr =
Discount.date_attr
        AND Sold.PID = Discount.PID
    GROUP BY MONTH(Sold.date_attr)) AS SMALL ON
MONTHS.`MONTH` = SMALL.`MONTH`
        LEFT JOIN
    (SELECT
        MONTH(Sold.date_attr) AS `MONTH`,
            SUM(quantity *
IF(ISNULL(Discount.discount_price), Product.retail_price,
Discount.discount_price)) AS `mediumCategory`,
            COUNT(Store.childcare_limit) AS `mediumCount`
```

```
    FROM
        Sold
    JOIN Product ON Product.PID = Sold.PID
    JOIN Store ON Store.childcare_limit
        AND Store.childcare_limit >= 10
        AND Store.childcare_limit <= 30
    LEFT OUTER JOIN Discount ON Sold.date_attr =
Discount.date_attr
        AND Sold.PID = Discount.PID
    GROUP BY MONTH(Sold.date_attr)) AS MEDIUM ON
MONTHS.`MONTH` = MEDIUM.`MONTH`
        LEFT JOIN
    (SELECT
        MONTH(Sold.date_attr) AS `MONTH`,
            SUM(quantity *
IF(ISNULL(Discount.discount_price), Product.retail_price,
Discount.discount_price)) AS `largeCategory`,
            COUNT(Store.childcare_limit) AS `largeCount`
```

```
    FROM
        Sold
    JOIN Product ON Product.PID = Sold.PID
    JOIN Store ON Store.childcare_limit
        AND Store.childcare_limit > 30
        AND Store.childcare_limit < 60
    LEFT OUTER JOIN Discount ON Sold.date_attr =
Discount.date_attr
        AND Sold.PID = Discount.PID
    GROUP BY MONTH(Sold.date_attr)) AS LARGE ON
MONTHS.`MONTH` = LARGE.`MONTH`
        LEFT JOIN
    (SELECT
        MONTH(Sold.date_attr) AS `MONTH`,
            SUM(quantity *
IF(ISNULL(Discount.discount_price), Product.retail_price,
Discount.discount_price)) AS `NoneCategory`,
            COUNT(Store.childcare_limit) AS `noneCount`
```

```
    FROM
        Sold
    JOIN Product ON Product.PID = Sold.PID
    JOIN Store ON Store.childcare_limit
        AND Store.childcare_limit = 0
    LEFT OUTER JOIN Discount ON Sold.date_attr =
Discount.date_attr
        AND Sold.PID = Discount.PID
    GROUP BY MONTH(Sold.date_attr)) AS EXTRALARGE ON
MONTHS.`MONTH` = EXTRALARGE.`MONTH`
ORDER BY MONTHs.`MONTH`;
```

- click ***Main Menu*** button, return to **Main Menu**.

Restaurant Impact on Category Sales

Task Decomp

Abstract Code

- User clicked on ***View Analyze Restaurant Impact on Sales by Categories*** button from **Main Menu**:
- Run the ***Analyze Restaurant Impact on Sales by Categories*** task.
- Display results of the following SQL:

```
SELECT Category, Store_Type, Quantity_Sold
FROM
     (SELECT unitsCategory AS Category, SUM(Quantity_Sold)
AS Quantity_Sold
            FROM
          (SELECT Category.category_name AS unitsCategory,
Store.has_restaurant, SUM(quantity) AS Quantity_Sold
          FROM Sold
          JOIN Product ON Product.PID = Sold.PID
          JOIN Store ON Sold.storeID = Store.storeID
          JOIN BelongTo ON BelongTo.PID = Sold.PID
          JOIN Category ON Category.category_name =
BelongTo.category_name
          WHERE
               MONTH(Sold.date_attr) = '$month' AND
YEAR(Sold.date_attr) = '$year'
          GROUP BY Category.category_name,
Store.has_restaurant) AS Q1
     GROUP BY unitsCategory) AS Q2
```

```
JOIN
          (SELECT Category.category_name AS
Store_TypeCategory, Store.has_restaurant AS Store_Type,
SUM(quantity) AS Store_TypeUnits
                        FROM Sold
                        JOIN Product ON Product.PID =
Sold.PID
          JOIN Store ON Sold.storeID = Store.storeID
          JOIN BelongTo ON BelongTo.PID = Sold.PID
          JOIN Category ON Category.category_name =
BelongTo.category_name
        WHERE
          MONTH(Sold.date_attr) = '$month' AND
YEAR(Sold.date_attr) = '$year'
        GROUP BY Category.category_name,
Store.has_restaurant) AS Q3 ON Category =
Store_TypeCategory AND Quantity_Sold = Store_TypeUnits
ORDER BY Category ASC;
```

- User clicks ***Main Menu*** button click run Display Main Menu task.

Advertising Campaign Analysis

Task Decomp

Abstract Code

- User clicked on ***View Analyze Sales during Advertising Campaigns*** button from **Main Menu**:
- Run the ***Analyze Sales during Advertising Campaigns*** task.
- Display results of the following SQL:

```
SELECT s.PID, p.product_name,
SUM(s.quantity) AS soldincamp
FROM
(Sold s
JOIN HasAdCamp c
ON s.date_attr = c.date_attr)
JOIN Product p
ON s.PID = p.PID
GROUP BY s.PID;

SELECT s.PID, p.product_name,
SUM(s.quantity) AS soldoutcamp
FROM
(
Sold s
JOIN Product p
ON s.PID = p.PID
)
LEFT JOIN HasAdCamp c USING (date_attr)
WHERE c.date_attr IS NULL
GROUP BY s.PID;
```

- User click ***Main Menu*** button to display **Main Menu** task.