Table of Contents

Login

Abstract Code

- User enters values for $username and $password
- If $username or $password are blank then:
  - Display error message on **Login** form.
- Else check for user record:
  - When the user clicks the ***Login*** button.

```
SELECT Password FROM 'User' WHERE Username = '$username';
```

  - If User.password = '$password':
    - Navigate to **Main Menu** form.
  - Else display error message on **Login** form.

Dashboard/MainMenu

Abstract Code

- Display **Main Menu** form after successful login.
- Display number of stores.

```
SELECT COUNT(*) FROM Store;
```

- Display number of Grand Showcase stores.

```
SELECT COUNT(*) FROM Store WHERE WillsGrandShowcase = TRUE;
```

- Check the type of user.

```
SELECT Type FROM 'User' WHERE Username = '$username';
```

- If User.Type = 'Store Manager':
    - Display number of stores that can be viewed.

```
SELECT COUNT(*) FROM Manages WHERE Username = '$username';
```

    - Add link to **Store Information** form.
    - Navigate to **Store Information** form when link is clicked.
    - Display all store information for stores managed by '$username'.

```
SELECT * FROM Store INNER JOIN Manages On Store.StoreID = Manages.StoreID;
```

- Display number of manufacturers.

```
SELECT COUNT(*) FROM Manufacturer;
```

- Display number of products.

```
SELECT COUNT(*) FROM Product;
```

- Display number of special savings days.

```
SELECT COUNT(*) FROM 'Business Day' WHERE SavingsDay = TRUE;
```

- Display buttons for ***Report 1, Report 2, Report 3, Report 4, Report 5, Report 6, Report 7, Report 8*** and ***Report 9***.
- When user clicks on a ***Report*** button to navigate to the corresponding **Report** form.
- When user clicks on the ***Logout*** button, navigate to the **Login** form.

Update Population

Abstract Code

- When user clicks the ***Update Population*** button from the **Main Menu** form

```
SELECT Type FROM 'User' WHERE Username = '$username';
```

- If User.Type != 'Marketing':
    - Display error message on the **Main Menu** form
- Else:
    - Navigate to **Update Population** form.
    - User enters values for $city and $state.
    - When user clicks the ***Display Population*** button

```
SELECT Population FROM 'User' WHERE CityName = '$city' and State = '$state';
```

o If User.Population is invalid:
- Display error message on the **Update Population** form.

o Else:
- Display the User.Population value on the **Update Population** form.
- User enters a new value for $population and clicks the ***Save*** button.
- If $population value is invalid:
    - Display error message the **Update Population** form.
- Else update the value in 'City':

```
UPDATE 'City' SET Population = '$population' WHERE CityName = '$city' and State = '$state';
```

- Display the updated $population value.

Report 1

- Get list of manufacturers

```
SELECT Name FROM Manufacturer;
```

- {for each manufacturer_name}
    - Find all PIDs and their prices from the manufacturer

```
SELECT UNIQUE PID, Price FROM Product WHERE Mname =
$manufacturer_name;
```

- Use price data for products to calculate average, min, max prices
- Sort manufacturers by average price descending
- Filter to just the top 100 manufacturers (based on avg price).
- Display summary price data (number of products, avg/min/max price) per manufacturer
- {if drill-down:}
    - Get Product Data for manufacturer (product ID, name, category, retail price)

```
SELECT UNIQUE PID, Product.Name, Price
FROM Product
WHERE Mname = $manufacturer_name
ORDER BY Price;
```

    - {for each PID}
        - Get Categories

```
SELECT Name
FROM IsInCategory
Where PID=$PID;
```

    - display summary data (number of products, avg/min/max price) , as well as product data (price, categories)

Report 2

- Get category list sorted by category name ascending

```
SELECT Name
From Category
ORDER BY Name ASC;
```

- {For each Name}
    - Pull all products and retail prices

```
SELECT Price
From Product, IsInCategory
WHERE IsInCategory.Name = $Name AND IsInCategory.PID=Product.PID;
```

- calculate avg/mins/max Prices
- fill out display with number of products, min prices, max prices, and avg prices (sorted by category name ascending)

Report 3

- Find product PID of all ‘couches and sofas’

```
SELECT p.PID, p.Name, p.Price
FROM Product AS p, IsInCategory AS c
WHERE c.Name=’couches and sofas’ AND p.PID = c.PID;
```

- {for each PID}
    - Find all dates and quantities of sales that are NOT on sales days

```
SELECT Quantity
FROM Sold,  `Discounted On`
WHERE Sold.PID = $PID AND Sold.Date NOT IN (
   SELECT Date
   FROM  `Discounted On`
   WHERE PID = $PID);
```

    - Find all sales and prices that ARE on sales days

```
SELECT Quantity, DiscountPrice
FROM Sold,  `Discounted On`
WHERE Sold.PID=$PID and Sold.Date IN (
   SELECT Date
   FROM  `Discounted On`
   Where PID = $PID
);
```

    - calculate revenue (Use product price for all non-discount sales, and discount price for all discount sales)
    - apply .75 rule to all discount days, and adjust price to retail price
    - Filter results to only predicted revenue differences +/- $5000
- Sort into descending order
- Display discount data, revenue, and prediction results

Report 4

- User clicked on ***Report 4*** button from **Dashboard**:
- Run the **Report 4** task
- Find all States using the City.State; Display all states in dropdown.

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

- Display the result of the below query in a table with selected state $State.

```
// Merge city name and address data
Select t.storeID, a.CityName, a.Address, year, t.TotalRevenue
// calculate revenue per year per store
FROM (Select year(t.date) AS year, t.storeID, sum(t.TotalRevenue) AS TotalRevenue
// apply Special Sale Days
FROM ((Select t.date, t.storeId, t.Revenue*(1-b.PercentDiscount) AS TotalRevenue
//calculate total revenue per day per store
FROM (Select g.Date, g.StoreID, sum(g.Revenue) AS Revenue FROM ((Select t.PID,
t.Date,t.StoreID,t.Quantity*t.Price AS Revenue
//calculate revenue per item per day including discount days
FROM (SELECT p.PID, s.Date, st.StoreID, s.Quantity, p.Price FROM Product AS p JOIN
Sold AS s JOIN Store AS st ON p.PID = s.PID AND s.StoreID = st.StoreID WHERE
st.State = $State) AS t LEFT JOIN
(SELECT d.PID, d.Date, d.Discountprice FROM `Discounted On` AS d) as g on t.PID =
g.PID and t.Date = g.Date WHERE g.Discountprice is NULL)
UNION
(Select t.PID, t.Date, t.StoreID, t.Quantity*g.Discountprice AS Revenue FROM (SELECT
p.PID, s.Date, st.StoreID, s.Quantity, p.Price FROM Product as p JOIN SOLD AS s JOIN
Store AS st ON p.PID = s.PID AND s.StoreID = st.StoreID WHERE st.State = $State) as t
LEFT JOIN
(SELECT d.PID, d.Date, d.DiscountPrice FROM `Discounted On` AS d) as g ON t.PID =
g.PID AND t.Date = g.Date WHERE g.Discountprice NOT NULL)) AS g GROUPBY
g.Date, g.StoreID) AS t LEFT JOIN `Business Day`as b on t.date = b.date)) as t
// group by year and store
GROUPBY year, t.storeid
//order by year and revenue
ORDERBY year asc, TotalRevenue desc) AS t LEFT JOIN Store AS a ON t.storeID =
a.storeID;
```

- User clicks on ***Back to Dashboard*** button to return to **Dashboard**

Report 5

- User clicked on ***Report 5*** button from **Dashboard**:
- Run the **Report 5** task
- Find the quantity of items sold, average, and quantity sold on “02-02” in the outdoor furniture category for each year using the IsInCategory.name;

```
SELECT q.year, q.total, q.average, a.groundhogs FROM
// Calculate total and average sales per year in the “Outdoor Furniture” category
                        (SELECT YEAR(s.Date) AS year, SUM(s.Quantity) AS total,
                        SUM(s.Quantity)/365 AS average
                        FROM Sold as S
                        LEFT JOIN IsInCategory AS c
                        ON s.PID = c.PID
                        WHERE c.CatName = "Outdoor Furniture"
                        GROUP BY(year)
                        ORDER BY Year ASC
                        ) AS q
LEFT JOIN
// Calculate sales per year in the “Outdoor Furniture” category on “02-02”
                        (SELECT YEAR(s.Date) AS year, SUM(s.Quantity) as groundhogs
                        FROM Sold AS s
                        LEFT JOIN IsInCategory AS c
                        ON s.PID = c.PID
                        WHERE c.CatName = "Outdoor Furniture" AND s.Date LIKE
                        '____-02-02'
GROUP BY year
ORDER BY Year ASC) AS a on a.year = q.year;
```

- Display results in table.
- User clicks on ***Back to Dashboard*** button to return to **Dashboard**

Report 6

- User clicked on ***Report 6*** button from **Dashboard**
- User selects $year and $month from the **Dropdown**
- Display the category name and total quantity per state

```
WITH VolumePerCategory(category, state, volume) AS (

SELECT cat.Name, c.State, SUM(s.Quantity)
FROM Sold s
LEFT JOIN product p ON p.PID = s.PID
```

```
LEFT JOIN IsInCategory iic ON p.PID = iic.PID
LEFT JOIN Category cat ON cat.Name = iic.CatName
LEFT JOIN `Business Day` bd ON bd.date = s.date
```

- Find all the dates for the requested month

```
LEFT JOIN (
SELECT Date
FROM `Business Day`
WHERE Date LIKE CONCAT($year, '/', $month, '%')) d ON (bd.Date =
d.Date)
)
```

- Filter the highest volume

```
SELECT Category, State, MAX(volume)
FROM VolumePerCategory
GROUP BY Volume, Category, State
ORDER BY Category
```

Report 7

- User clicked on ***Report 7*** button from **Dashboard**
- Calculate the revenue

```
WITH Revenue AS (
SELECT
CASE
WHEN bd.PercentDiscount IS NOT NULL
THEN p.price * (1 - bd.PercentDiscount) * sld.Quantity
WHEN bd.PercentDiscount IS NULL AND do.DiscountAmount is NOT NULL
THEN P.price - do.DiscountAmount * sld.Quantity
ELSE p.Price * sld.Quantity
END
)
```

- Create city size categories and display the revenue for each year and city size category in a tabular format

```
SELECT
YEAR(s.Date) AS Year,
SUM(CASE WHEN c.Population  < 3 700 000 THEN Limit END) AS Small,
SUM(CASE WHEN c.Population  BETWEEN 3 700 000 AND 6 700 000 THEN Limit END) AS Medium,
SUM(CASE WHEN c.Population  BETWEEN 6 700 000 AND 9 000 000 THEN Limit END) AS Large,
SUM(CASE WHEN c.Population  > 9 000 000 THEN Limit END) AS 'Extra Large'
)
FROM City c
LEFT JOIN PopulationCategory pc ON pc.Population = c.Population
LEFT JOIN Store s ON c.Name = s.CityName
```

```
LEFT JOIN Sold sld ON sld.StoreID = s.StoreID
LEFT JOIN Product p ON sld.PID = p.PID
LEFT JOIN `Business Day` bd ON bd.Date = sld.Date
LEFT JOIN `Discounted On` do ON do.PID = p.PID AND do.Date = bd.Date
GROUP BY Year
ORDER BY Year, pc.Category DESC
```

Report 8

- User clicked on ***Report 8*** button from **Dashboard**:
- Run ***Report 8*** task: query performance of Will's Grand Showcase stores compared to non-Showcase stores, where $User is the current user's Username
- Header:

```
SELECT
        CASE WHEN s.WillsGrandShowcase = 1 THEN "Will's Grand Showcase
Stores"
                WHEN s.WillsGrandShowcase = 0 THEN "Non-Showcase Stores"
                END AS "Store Type",
        COUNT(DISTINCT s.StoreID)
FROM Store
```

- Report:

```
WITH StoreRevenue AS (
        SELECT YEAR(sld.date) Year,
        st.StoreID,
        st.willsgrandshowcase,
        CASE WHEN bd.percentdiscount IS NOT NULL
```

```
                        THEN p.price * (1-bd.percentdiscount) * sld.quantity
                WHEN bd.percentdiscount IS NULL AND do.discountamount IS NOT
        NULL
                        THEN p.price - do.discountamount  * sld.quantity
                ELSE p.price * sld.quantity
                END AS Revenue
        FROM Store st
        LEFT JOIN Sold sld
                ON st.StoreID = sld.StoreID
        LEFT JOIN  Product p
                ON sld.PID = p.PID
        LEFT JOIN `Discounted On` do
                ON sld.date = do.date
                AND p.PID = do.PID
                AND st.StoreID = do.StoreID
        LEFT JOIN `Business Day` bd
                ON sld.date = bd.date
        WHERE st.willsgrandshowcase = 1
),
GrandShowcaseStats AS (
        SELECT year,
                MIN(revenue) gs_min_rev,
                AVG(revenue) gs_avg_rev,
                MAX(revenue) gs_max_rev
        FROM StoreRevenue
        WHERE willsgrandshowcase = 1
        GROUP BY year
),
NonShowcaseStats AS (
        SELECT year,
                MIN(revenue) ngs_min_rev,
                AVG(revenue) ngs_avg_rev,
                MAX(revenue) ngs_max_rev
        FROM StoreRevenue
        WHERE willsgrandshowcase = 0
        GROUP BY year
)
SELECT gss.year,
        gss.gs_min_rev as `Grand Showcase Minimum Revenue`,
        gss.gs_avg_rev as `Grand Showcase Average Revenue`,
```

```
        gss.gs_max_rev as “Grand Showcase Maximum Revenue”,
        nss.ngs_min_rev as “Non Grand Showcase Minimum Revenue”,
        nss.ngs_avg_rev as “Non Grand Showcase Average Revenue”,
        nss.ngs_max_rev as “Non Grand Showcase Maximum Revenue”
FROM GrandShowcaseStats gss
LEFT JOIN NonShowcaseStats nss
        ON gss.year = nss.year
ORDER BY gss.year;
```

- User clicks on ***Back to Dashboard*** button to return to **Dashboard**

Report 9

- User clicked on ***Report 9*** button from **Dashboard**:
- Run ***Report 9*** task: query category performance of Will’s Grand Showcase stores compared to non-Showcase stores

```
WITH product_sales_by_showcase AS (
SELECT p.PID,
SUM(CASE WHEN st.willsgrandshowcase = 1 THEN sld.quantity ELSE 0 END)
gs_sold,
SUM(CASE WHEN st.willsgrandshowcase = 1 THEN sld.quantity ELSE 0 END)
ngs_sold
FROM Product p
LEFT JOIN Sold sld
        ON p.PID = sld.PID
LEFT JOIN Store st
        ON sld.StoreID = st.StoreID
GROUP BY p.PID
)
SELECT c.Name AS Category,
        SUM(COALESCE(gs_sold,0)) AS “Grand Showcase Qty”,
        SUM(COALESCE(ngs_sold,0)) AS “Non-Showcase Qty”,
        (SUM(COALESCE(gs_sold,0)) - SUM(COALESCE(ngs_sold,0))) AS Difference
FROM Category c
LEFT JOIN IsInCategory iic
        ON c.Name = iic.CatName
LEFT JOIN product_sales_by_showcase pss
```

```
        ON iic.PID = pss.PID
GROUP BY c.Name
ORDER BY (SUM(COALESCE(gs_sold,0)) - SUM(COALESCE(ngs_sold,0))) DESC,
        c.Name ASC;
```

- Include links for each category to display a drill down when selected
- Run ***Report 9*** drill down for a link: query product performance of Will's Grand Showcase stores compared to non-Showcase stores, where $Category is the selected category

```
WITH product_sales_by_showcase AS (
SELECT p.PID,
p.Name,
SUM(CASE WHEN st.willsgrandshowcase = 1 THEN sld.quantity ELSE 0 END)
gs_sold,
SUM(CASE WHEN st.willsgrandshowcase = 1 THEN sld.quantity ELSE 0 END)
ngs_sold
FROM Product p
LEFT JOIN Sold sld
        ON p.PID = sld.PID
LEFT JOIN Store st
        ON sld.StoreID = st.StoreID
GROUP BY p.PID, p.Name
)
SELECT p.PID,
        p.Name
        SUM(COALESCE(gs_sold,0)) AS “Grand Showcase Qty”,
        SUM(COALESCE(ngs_sold,0)) AS “Non-Showcase Qty”,
        (SUM(COALESCE(gs_sold,0)) - SUM(COALESCE(ngs_sold,0))) AS Difference
FROM Category c
LEFT JOIN IsInCategory iic
        ON c.Name = iic.CatName
LEFT JOIN product_sales_by_showcase pss
        ON iic.PID = pss.PID
WHERE c.Name = $Category
GROUP BY p.PID, p.Name
ORDER BY (SUM(COALESCE(gs_sold,0)) - SUM(COALESCE(ngs_sold,0))) DESC,
```

```
        p.PID ASC
LIMIT 5

UNION

SELECT p.PID,
        p.Name
        SUM(COALESCE(gs_sold,0)) AS “Grand Showcase Qty”,
        SUM(COALESCE(ngs_sold,0)) AS “Non-Showcase Qty”,
        (SUM(COALESCE(gs_sold,0)) - SUM(COALESCE(ngs_sold,0))) AS Difference
FROM Category c
LEFT JOIN IsInCategory iic
        ON c.Name = iic.CatName
LEFT JOIN product_sales_by_showcase pss
        ON iic.PID = pss.PID
WHERE c.Name = $Category
GROUP BY p.PID, p.Name
ORDER BY (SUM(COALESCE(gs_sold,0)) - SUM(COALESCE(ngs_sold,0))) ASC,
        p.PID ASC
LIMIT 5;
```

- User clicks on ***Back to Dashboard*** button to return to **Dashboard**