Phase 3 SQL Compilation | CS 6400 – Spring 2019 | Team 07

This document is a compilation of all SQL used in project. All SQL used in the project is taken from files in the `app/sql` *folder. Path/file names used in the project are shown in green, and the SQL contained in them follows, while path/file names not used in the project are shown in red, and no SQL is shown.*

app/sql/01. Display Main Menu.sql:

```
SELECT COUNT(storeNumber) AS 'Number of Stores' FROM Store;
SELECT COUNT(manufacturerName) AS 'Number of Manufacturers' FROM Manufacturer;
SELECT COUNT(PID) AS 'Number of Products' FROM Product;
SELECT COUNT(categoryName) FROM Category;
SELECT COUNT(emailAddress) AS 'Number of Managers' FROM Manager;
SELECT COUNT(DISTINCT emailAddress) AS 'Number of Active Managers' FROM Manages;
SELECT COUNT(*) FROM (SELECT S.storeNumber AS store, M.storeNumber AS managed FROM Store S
        LEFT OUTER JOIN Manages M ON S.storeNumber = M.storeNumber WHERE
        ISNULL(M.storeNumber)) C;
SELECT COUNT(*) FROM StoreSellsProduct;
SELECT COUNT(*) FROM GoesOnSale;
SELECT COUNT(*) FROM Holiday;
SELECT DATABASE();
SELECT SYSTEM_USER();
```

app/sql/02 Edit Holiday Data.sql:

```
SELECT holidayName, holidayDate FROM Holiday ORDER BY holidayDate DESC;
INSERT INTO Holiday (holidayDate, holidayName) VALUES (%s, %s);
UPDATE Holiday SET holidayName = %s WHERE holidayDate = %s;
```

app/sql/03. Edit Manager Profile.sql:

```
SELECT managerName, emailAddress FROM Manager ORDER BY managerName;
INSERT INTO Manager(emailAddress, managerName) VALUES (%s, %s);
UPDATE Manager SET managerName = %s WHERE emailAddress = %s;
DELETE FROM Manages WHERE emailAddress = %s;
SELECT storeNumber, emailAddress FROM Manages WHERE emailAddress = %s;
DELETE FROM Manager WHERE emailAddress = %s;
SELECT COUNT(emailAddress) FROM Manages WHERE emailAddress = %s;
DELETE FROM Manager WHERE emailAddress = %s;
```

app/sql/04. Assign Stores.sql:

```
SELECT storeNumber FROM Manages WHERE emailAddress = %s ORDER BY storeNumber;
SELECT emailAddress FROM Manages WHERE storeNumber = %s ORDER BY emailAddress;
INSERT INTO Manages(storeNumber, emailAddress) VALUES(%s, %s);
DELETE FROM Manages WHERE emailAddress = %s AND storeNumber = %s;
SELECT GROUP_CONCAT(S.storeNumber SEPARATOR ', ') FROM Store S WHERE S.storeNumber NOT IN
        (SELECT M.storeNumber FROM Manages M);
SELECT storeNumber, managerName, Manages.emailAddress FROM Manages JOIN Manager ON
        Manages.emailAddress = Manager.emailAddress ORDER BY Manages.storeNumber,
        managerName;
SELECT storeNumber, storeNumber FROM Store ORDER BY storeNumber;
SELECT emailAddress, CONCAT(managerName, ' (', emailAddress, ')') FROM Manager ORDER BY
        managerName;
```

app/sql/05. Update City Population.sql:

```
SELECT cityName, state, population FROM City ORDER BY state, cityName;
SELECT CONCAT(cityName, '|', state) as `cityValue`, CONCAT(cityName, ', ', state) as `city` FROM City
        ORDER BY state, cityName;
UPDATE City SET Population = %s WHERE cityName = %s AND State = %s;
```

app/sql/06. Display Manufacturer Summary.sql:

```
SELECT
   manufacturerName,
   COUNT(PID),
   ROUND(AVG(price), 2),
   ROUND(MIN(price), 2),
   ROUND(MAX(price), 2)
FROM Product
GROUP BY manufacturerName ORDER BY AVG(price) DESC LIMIT 100;
```

07a. Display Manufacturer Drill-Down Details.sql:

```
SELECT maximumDiscount
FROM Manufacturer
WHERE manufacturerName = %s;
```

app/sql/07b. Display Manufacturer Drill-Down Details.sql:

```
SELECT
Product.PID AS `PID`,
productName AS `Name`,
GROUP_CONCAT(DISTINCT categoryName SEPARATOR ', ') AS `Categories`,
ROUND(price, 2)
FROM Product, CategorizedBy
WHERE
    CategorizedBy.PID = Product.PID AND manufacturerName = %s
GROUP BY Product.PID
ORDER BY price DESC;
```

app/sql/07c. Display Manufacturer Drill-Down Details.sql:

```
SELECT
   maximumDiscount,
   COUNT(PID),
   ROUND(AVG(price), 2),
   ROUND(MIN(price), 2),
   ROUND(MAX(price), 2)
FROM Product P, Manufacturer M
WHERE P.manufacturerName = M.manufacturerName AND M.manufacturerName = %s;
```

app/sql/08. Category Summary.sql:

```
SELECT C.categoryName AS Category, COUNT(CB.PID) AS `Number of Products`,
COUNT(DISTINCT P.manufacturerName) AS `Number of Manufacturers`, ROUND(AVG(P.price), 2) AS
        `Average Price`
FROM Category C JOIN CategorizedBy CB ON C.categoryName = CB.categoryName JOIN Product P ON
        CB.PID = P.PID
GROUP BY C.categoryName ORDER BY C.categoryName;
```

app/sql/09. Display Actual vs Predicted GPS.sql [NOT USED IN PROJECT]

app/sql/09.1. Display Actual vs Predicted GPS.sql [NOT USED IN PROJECT]

app/sql/09.2. Display Actual vs Predicted GPS.sql:

```
SELECT
   `Product ID`,
   `Product Name`,
   `Retail Price`,
   SUM(quantity) AS 'Total Units Sold',
   SUM(`Sale Quantity`) AS 'Total Units Sold at Discount',
   SUM(`Retail Quantity`) AS 'Total Units Sold at Retail',
   ROUND(SUM(`Transaction Amount`), 2) AS 'Actual Revenue',
   ROUND(SUM(`Predicted Amount`), 2) AS 'Predicted Revenue',
   ROUND(SUM(InnerSelect.Difference), 2) AS 'Difference'
FROM
   (SELECT
      S.PID AS 'Product ID',
      P.productName AS 'Product Name',
      ROUND(P.price, 2) AS 'Retail Price',
      transactionDate,
      IF(ISNULL(G.salePrice), 0, quantity) AS 'Sale Quantity',
      IF(ISNULL(G.salePrice), quantity, 0) AS 'Retail Quantity',
      quantity * IF(ISNULL(G.salePrice), P.price, G.salePrice) AS 'Transaction Amount',
      IF(ISNULL(G.salePrice), quantity * P.price, 0.75 * quantity * P.price) AS 'Predicted Amount',
      quantity * IF(ISNULL(G.salePrice), P.price, G.salePrice) -
         IF(ISNULL(G.salePrice), quantity * P.price, 0.75 * quantity * P.price) AS 'Difference',
      quantity
   FROM StoreSellsProduct S
   JOIN CategorizedBy C
      ON S.PID = C.PID AND C.categoryName = "GPS"
   JOIN Product P
      ON S.PID = P.PID
   LEFT OUTER JOIN GoesOnSale G
      ON S.PID = G.PID AND transactionDate = saleDate
   ) AS InnerSelect
GROUP BY `Product ID`, `Product Name`, `Retail Price`
HAVING ABS(Difference) > 5000
ORDER BY `Difference` DESC, `Product ID`;
```

app/sql/10. Display Store Revenue by Year and State.sql:

```
SELECT
	Store.storeNumber,
	Store.streetAddress,
	City.cityName,
	YEAR(StoreSellsProduct.transactionDate) AS Year,
	ROUND(SUM(IF(StoreSellsProduct.transactionDate = GoesOnSale.saleDATE AND
		StoreSellsProduct.PID = GoesOnSale.PID,GoesOnSale.salePrice, Product.price)
		* StoreSellsProduct.quantity),2) AS Revenue
FROM
	StoreSellsProduct
LEFT OUTER JOIN
	GoesOnSale
ON
	StoreSellsProduct.transactionDate = GoesOnSale.saleDATE AND
		StoreSellsProduct.PID = GoesOnSale.PID,
	Product,
	Store,
	City
WHERE
	City.state = %s
	AND Store.state = %s
	AND Store.cityName = City.cityName
	AND StoreSellsProduct.storeNumber = Store.storeNumber
	AND StoreSellsProduct.PID = Product.PID
GROUP BY
	Store.storeNumber,
	Store.streetAddress,
	City.cityName,
	YEAR(StoreSellsProduct.transactionDate)
ORDER BY
	Year,
	Revenue DESC;
```

app/sql/10.1. Display Store Revenue by Year and State.sql:

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

app/sql/11. Display Air Conditioners on Groundhog Day.sql:

```
SELECT
	YEAR(StoreSellsProduct.transactionDate) AS Year,
	SUM(StoreSellsProduct.quantity) AS `Total AC Items Sold`,
	ROUND(SUM(StoreSellsProduct.quantity)/365, 0) AS
		`Average (Rounded) Number of Units Sold/Day`,
	GH.Total AS `Total Units Sold on GroundHog Day`
FROM
	StoreSellsProduct,
	CategorizedBy,
	(
		SELECT
			SUM(StoreSellsProduct.quantity) AS Total,
			YEAR(StoreSellsProduct.transactionDate) AS Year
		FROM
			StoreSellsProduct,
			CategorizedBy
		WHERE
			Categorizedby.categoryName = 'Air Conditioner' AND
			StoreSellsProduct.PID = Categorizedby.PID AND
			DATE_FORMAT(StoreSellsProduct.transactionDate,'%m-%d') = '02-02'
		GROUP BY
		YEAR(StoreSellsProduct.transactionDate)
	) AS GH
WHERE
	Categorizedby.categoryName = 'Air Conditioner' AND
	StoreSellsProduct.PID = Categorizedby.PID AND
	GH.Year = YEAR(StoreSellsProduct.transactionDate)
GROUP BY
	YEAR(StoreSellsProduct.transactionDate),
	GH.Total
ORDER BY
	YEAR ASC;
```

app/sql/12. Display State Volume by Category.sql [NOT USED IN PROJECT]

app/sql/12.1 Display State Volume by Category.sql:

```
SELECT DISTINCT tDate
        FROM
          (SELECT
            DATE_FORMAT(StoreSellsProduct.transactionDate,'%m-%Y') AS `tDate`
          FROM
            StoreSellsProduct) AS D
        ORDER BY
          SUBSTRING(tDate, 4, 4) DESC, SUBSTRING(tDate, 1, 2) DESC;
```

app/sql/12.2 Display State Volume by Category.sql:

```
SELECT
	Category, State, Units
FROM
(
	SELECT
		unitsCategory AS Category, MAX(units) AS Units
	FROM
	(
		SELECT
			CAT.categoryName AS unitsCategory,
			City.state, SUM(quantity) AS units
		FROM
			StoreSellsProduct SSP
		JOIN Product P ON P.PID = SSP.PID
		JOIN Store S ON SSP.storeNumber = S.storeNumber
		JOIN City ON City.cityName = S.cityName AND City.state = S.state
		JOIN CategorizedBy CB ON CB.PID = SSP.PID
		JOIN Category CAT ON CAT.categoryName = CB.categoryName
		WHERE
			MONTH(SSP.transactionDate) = %s AND
			YEAR(SSP.transactionDate) = %s
		GROUP BY CAT.categoryName, City.state
	) AS Q1

	GROUP BY unitsCategory
) AS Q2
JOIN
(
	SELECT
		CAT.categoryName AS stateCategory,
		City.state AS State,
		SUM(quantity) AS stateUnits
		FROM
			StoreSellsProduct SSP
		JOIN
			Product P ON P.PID = SSP.PID
		JOIN
			Store S ON SSP.storeNumber = S.storeNumber
		JOIN
			City ON City.cityName = S.cityName AND City.state = S.state
		JOIN
			CategorizedBy CB ON CB.PID = SSP.PID
		JOIN
			Category CAT ON CAT.categoryName = CB.categoryName
		WHERE
			MONTH(SSP.transactionDate) = %s AND
```

```
                    YEAR(SSP.transactionDate) = %s
                GROUP BY
                    CAT.categoryName, City.state
) AS Q3
    ON Category = stateCategory AND Units = stateUnits
ORDER BY Category;
```

app/sql/12.3 Display State Volume by Category.sql:

```
SELECT
    Store.storeNumber AS `Store Number`,
    streetAddress AS `Address`,
    cityName AS City,
    IFNULL(managerName, '[No manager assigned]') AS Manager,
    Manager.emailAddress AS Email
FROM
    Store
LEFT JOIN
    Manages
    ON Manages.storeNumber = Store.storeNumber
LEFT JOIN
    Manager
    ON Manager.emailAddress = Manages.emailAddress
WHERE
    Store.state= %s
ORDER BY
    Store.storeNumber, managerName;
```

app/sql/13. Display Annual Average Revenue by City Population.sql [NOT USED IN PROJECT]

app/sql/13.1. Display Annual Total Revenue by City Population.sql [NOT USED IN PROJECT]

app/sql/13.2. Display Annual Average Revenue by City Population.sql:

```
SELECT
	YEARS.`Year`,

	ROUND(IFNULL(smallCategory, 0), 2) AS `Small Category`,
	ROUND(IFNULL(mediumCategory, 0), 2) AS `Medium Category`,
	ROUND(IFNULL(largeCategory, 0), 2) AS `Large Category`,
	ROUND(IFNULL(extraLargeCategory, 0), 2) AS `Extra Large Category`,

	ROUND(IFNULL(smallCategory/smallCount, 0), 2) AS `Small Category Avg`,
	ROUND(IFNULL(mediumCategory/mediumCount, 0), 2) AS `Medium Category Avg`,
	ROUND(IFNULL(largeCategory/largeCount, 0), 2) AS `Large Category Avg`,
	ROUND(IFNULL(extraLargeCategory/extraLargeCount, 0), 2) AS `Extra Large Category Avg`
FROM
	(
		SELECT
			DISTINCT YEAR(transactionDate) AS `Year`
		FROM
			StoreSellsProduct SSP
	) AS YEARS
LEFT JOIN
	(
		SELECT
			YEAR(transactionDate) AS `Year`,
			ROUND(
				SUM(quantity *
				IF(ISNULL(GOS.salePrice), P.price, GOS.salePrice)),
				2) AS `smallCategory`,
			COUNT(C.cityName) AS `smallCount`
		FROM
			StoreSellsProduct SSP
		JOIN
			Product P
		ON
			P.PID = SSP.PID
		JOIN
			Store S
		ON
			S.storeNumber = SSP.storeNumber
		JOIN
			City C
		ON
			C.cityName = S.cityName
			AND C.state = S.state
			AND C.population < 3700000
		LEFT OUTER JOIN
			GoesOnSale GOS
```

```
                ON
                        SSP.transactionDate = GOS.saleDate
                        AND SSP.PID = GOS.PID
                GROUP BY
                        YEAR(transactionDate)
        ) AS SMALL
ON
        YEARS.`Year` = SMALL.`Year`
LEFT JOIN
        (
                SELECT
                        YEAR(transactionDate) AS `Year`,
                        ROUND(
                                SUM(quantity * IF(ISNULL(GOS.salePrice),
                                        P.price, GOS.salePrice)),
                                2) AS `mediumCategory`,
                        COUNT(C.cityName) AS `mediumCount`
                FROM
                        StoreSellsProduct SSP
                JOIN
                        Product P
                ON
                        P.PID = SSP.PID
                JOIN
                        Store S
                ON
                        S.storeNumber = SSP.storeNumber
                JOIN
                        City C
                ON
                        C.cityName = S.cityName
                        AND C.state = S.state
                        AND C.population < 6700000
                        AND C.population >= 3700000
                LEFT OUTER JOIN
                        GoesOnSale GOS
                ON
                        SSP.transactionDate = GOS.saleDate
                        AND SSP.PID = GOS.PID
                GROUP BY
                        YEAR(transactionDate)
        ) AS MEDIUM
ON
        YEARS.`Year` = MEDIUM.`Year`
LEFT JOIN
        (
                SELECT
                        YEAR(transactionDate) AS `Year`,
```

```
                ROUND(
                        SUM(quantity *
                                IF(ISNULL(GOS.salePrice), P.price, GOS.salePrice)),
                        2) AS `largeCategory`,
                COUNT(C.cityName) AS `largeCount`
        FROM
                StoreSellsProduct SSP
        JOIN
                Product P
        ON
                P.PID = SSP.PID
        JOIN
                Store S
        ON
                S.storeNumber = SSP.storeNumber
        JOIN
                City C
        ON
                C.cityName = S.cityName
                AND C.state = S.state
                AND C.population < 9000000
                AND C.population >= 6700000
        LEFT OUTER JOIN GoesOnSale GOS
        ON
                SSP.transactionDate = GOS.saleDate
                AND SSP.PID = GOS.PID
        GROUP BY
        YEAR(transactionDate)
    ) AS LARGE
ON
    YEARS.`Year` = LARGE.`Year`
LEFT JOIN
    (
        SELECT
                YEAR(transactionDate) AS `Year`,
                ROUND(
                        SUM(quantity
                                * IF(ISNULL(GOS.salePrice), P.price, GOS.salePrice)),
                        2) AS `extraLargeCategory`,
                COUNT(C.cityName) AS `extraLargeCount`
        FROM
                StoreSellsProduct SSP
        JOIN
                Product P
        ON
                P.PID = SSP.PID
        JOIN
                Store S
```

```
                ON
                        S.storeNumber = SSP.storeNumber
                JOIN City C
                ON
                        C.cityName = S.cityName
                        AND C.state = S.state
                        AND C.population >= 9000000
                LEFT OUTER JOIN
                        GoesOnSale GOS
                ON
                        SSP.transactionDate = GOS.saleDate
                        AND SSP.PID = GOS.PID
                GROUP BY
                        YEAR(transactionDate)
        ) AS EXTRALARGE
ON
        YEARS.`Year` = EXTRALARGE.`Year`
ORDER BY YEARS.`Year`;
```

app/sql/13b. Display Annual Total Revenue by City Population.sql [NOT USED IN PROJECT]

app/sql/14. Display Groundhog Day.sql:

```
SELECT
   YEAR(StoreSellsProduct.transactionDate) AS Year,
   SUM(StoreSellsProduct.quantity) AS `Total AC Items Sold`,
   ROUND(SUM(StoreSellsProduct.quantity)/365, 0) AS
      `Average (Rounded) Number of Units Sold/Day`,
   GH.Total AS `Total Units Sold on GroundHog Day`
FROM
   StoreSellsProduct,
   CategorizedBy,
   (SELECT
      SUM(StoreSellsProduct.quantity) AS Total,
      YEAR(StoreSellsProduct.transactionDate) AS Year
   FROM
      StoreSellsProduct,
      CategorizedBy
   WHERE
      Categorizedby.categoryName = 'air conditioner' AND
      StoreSellsProduct.PID = Categorizedby.PID AND
      DATE_FORMAT(StoreSellsProduct.transactionDate,'%m-%d') =
         '02-02'
   GROUP BY YEAR(StoreSellsProduct.transactionDate)
   ) AS GH
WHERE
   Categorizedby.categoryName = 'air conditioner' AND
   StoreSellsProduct.PID = Categorizedby.PID AND
   GH.Year = YEAR(StoreSellsProduct.transactionDate)
GROUP BY
   YEAR(StoreSellsProduct.transactionDate),
   GH.Total
ORDER BY YEAR ASC;
```