LIBR-554: Database Design

SQL QUERIES
&
PHYSICAL DESIGN

Submitted To:
Richard Arias-Hernandez

Submitted By:
Rishab Madaan : 63403331
& Kritika Arora : 63945489

Table of Contents

I. SQL QUERIES

1) ***User***: The prime user of this query will be the boutique owner, though it may also be used by the store manager.

Purpose: This query will be used for the purpose of analyzing which product has been sold the most in the last X days from a given date and the gross income earned from the sale. The result set is ordered based on the net sales in descending order and also states the numbers of products sold for each category.

Required Layout for the result set: PRODUCT_NAME, TOTAL_ORDER_SALE_AMOUNT (SUM(O.ORDER_AMOUNT)), TOTAL_ORDER_SALE_QUANTITY (COUNT(O.ORDER_ID))

SQL Statement

```
CREATE PROCEDURE GetProductSale (STARTDATE DATE, DAYS INT)
SELECT P.PRODUCT_NAME, SUM(O.ORDER_AMOUNT) AS TOTAL_ORDER_SALE_AMOUNT,
COUNT(O.ORDER_ID) AS TOTAL_ORDER_SALE_QUANTITY
FROM PRODUCT P
JOIN MANUFACTURED_PRODUCT AS MP
ON P.PRODUCT_ID = MP.PRODUCT_ID
JOIN `ORDER` O ON O.ORDER_ID = MP.ORDER_ID
WHERE (O.ORDER_DATE BETWEEN DATE_SUB(STARTDATE, INTERVAL DAYS DAY)
AND STARTDATE )
GROUP BY P.PRODUCT_NAME
ORDER BY SUM(O.ORDER_AMOUNT) DESC;

CALL GetProductSale("2020-10-01", 90);
```

Result Set : Example set with data for last 90 days from 2020-10-01

	PRODUCT_NAME	TOTAL_ORDER_SALE_AMOUNT	TOTAL_ORDER_SALE_QUANTITY
▶	LEHNGA	12600.00	2
	CHOLI	11900.00	2
	GHAGHRA	6000.00	1

2) ***User***: The prime user of this query will be the boutique owner.

Purpose: The purpose of this query is to help the owner review the top 5 raw materials used and the money spent on buying these raw materials. The result set is ordered by the total bill amount in descending order and includes only raw materials which have been bought between the current date and in the past 1 year. This is important in any business to generate income statements and control expenditure if needed.

Required Layout for the result set: RAW MATERIAL (RAW_MATERIAL_NAME), TOTAL BILL (SUM(BILL_AMOUNT))

SQL Statement

```
CREATE PROCEDURE GetRawMaterialCost ()
SELECT R.RAW_MAT_NAME AS 'RAW MATERIAL', SUM(B.BILL_AMOUNT) AS 'TOTAL BILL'
FROM RAW_MATERIAL R
JOIN BILL AS B ON R.RAW_MAT_ID = B.RAW_MAT_ID
WHERE (B.BILL_DATE BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
AND CURDATE() )
GROUP BY R.RAW_MAT_NAME
ORDER BY SUM(B.BILL_AMOUNT) DESC LIMIT 5;


CALL GetRawMaterialCost();
```

Result Set: The result set is limited to 5 results and we see that the most amount of money was spent on Flower Patch

RAW MATERIAL	TOTAL BILL
FLOWER PATCH	9100.00
COTTON CLOTH	2900.00
SYNTHETIC CLOTH	1200.00
LACE	1120.00
RIBBON	1058.00

3) ***User***: The owner and store managers will be the user of this query. There is no sensitive information returned in the RESULT SET.

Purpose: This query returns the average rating given by customers, the customer's name and his phone number when the average rating is less than or equal to a given rating parameter. This helps the owner and the store managers to understand the satisfaction level of different customers and helps classify customers between different categories (happy, unhappy, target customers).

Required layout for the result set: CUSTOMER_NAME, CUSTOMER_PHONE, AVERAGE RATING GIVEN BY CUSTOMER

SQL Statement

```
CREATE PROCEDURE CustomerRatingOrder (RATING INT)
SELECT CONCAT (C.CUST_FNAME,  ' ', C.CUST_LNAME) AS 'CUSTOMER NAME', C.CUST_PHONE,
AVG(CLT.CLIENT_RATING_OUT_OF_5)  ' AVERAGE RATING GIVEN BY CUSTOMER '
FROM CUSTOMER C
JOIN CLIENT_TESTIMONIAL CLT ON C.CUST_ID = CLT.CUST_ID
GROUP BY CLT.CUST_ID
HAVING AVG(CLT.CLIENT_RATING_OUT_OF_5) <= RATING
ORDER BY COUNT(CLT.CUST_ID);


CALL CustomerRatingOrder (3);
```

Result Set: Returns the result set with customers having average rating equal to or less than 3 which is passed as parameter ordered by the number of ratings given by a customer.

CUSTOMER NAME	CUST_PHONE	AVERAGE RATING GIVEN BY CUSTOMER
MALINI ARORA	9880897651	3.0000
KARISHMA CHABHRA	9220897651	2.0000
PREITY SINHA	9093397651	3.0000

4) ***User***: This query can be used both by the owner and the store managers of the boutique.

Purpose: This query helps to understand the most loyal customers and can also be used later on to incorporate some sort of point rewards which is commonly used by different businesses. The query returns the customers list who have their total amount of orders greater than the given threshold and who have ordered after a given date along with the sum of their orders. Both these are passed as parameters to the procedure.

Required layout for the result set: CUSTOMER_NAME, CUSTOMER_PHONE, SUM OF ALL CUSTOMER ORDERS, NUMBER OF ORDER BY CUSTOMER

SQL Statement

```
USE K_KHWABDAAH;
CREATE PROCEDURE CustomerOrders(AMOUNT NUMERIC, STARTDATE DATE)
SELECT CONCAT (C.CUST_FNAME, ' ', C.CUST_LNAME) AS 'CUSTOMER NAME', C.CUST_PHONE,
SUM(P.PAYMENT_AMOUNT) AS 'SUM OF ALL CUSTOMER ORDERS',
COUNT(P.CUS_ID) AS 'NUMBER OF ORDERS BY CUSTOMER'
FROM CUSTOMER C
JOIN PAYMENT P ON C.CUST_ID = P.CUS_ID
WHERE P.PAYMENT_DATE > STARTDATE
GROUP BY P.CUS_ID
```

```
HAVING SUM(P.PAYMENT_AMOUNT) > AMOUNT
ORDER  BY COUNT(P.CUS_ID) DESC, SUM(P.PAYMENT_AMOUNT) DESC;
```

```
CALL CustomerOrders (100, '2020-01-12');
```

Result Set: Returns the result set ordered first by maximum number of products bought by a customer and then by total sum of customer's order amount when the total order amount is greater than a given threshold for the orders whose payment have been made after a given date.

CUSTOMER NAME	CUST_PHONE	SUM OF ALL CUSTOMER ORDERS	NUMBER OF ORDERS BY CUSTOMER
KATRINA KALSI	9034897651	12500.00	2
RITA SHARMA	9990897651	5000.00	2
KAREENA VERMA	9090895551	8000.00	1
KARISHMA CHABHRA	9220897651	7000.00	1
RIA MAINI	9055897651	7000.00	1
KAVYA AGGARWAL	9770897651	6000.00	1
PREITY SINHA	9093397651	6000.00	1
PREETY ARORA	9094497651	5600.00	1
MALINI ARORA	9880897651	4000.00	1
PRIYA MAINI	9090888651	3000.00	1
POOJA GOYAL	9090997651	2500.00	1
RAGHIKA GUPTA	8890897651	2000.00	1
SAMARI SINHA	9090897651	1500.00	1

5) ***User***: This query can only be run the owner as it contains sensitive data regarding payroll information.

Purpose: This query helps to determine the current position of a certain employee and the latest monthly salary that is being paid to that employee.

Required layout for the result set: EMPLOYEE_FNAME, EMPLOYEE_LNAME, EMPLOYEE_PHONE, PAY_POSITION, PAY_AMOUNT

SQL Statement

```
CREATE PROCEDURE EmployeeData (EMPLOYEE_FNAME VARCHAR(20), EMPLOYEE_LNAME
VARCHAR(20))
SELECT E.EMP_FNAME, E.EMP_LNAME, E.EMP_PHONE, PHST.PAY_POSITION, P.PAY_AMOUNT
FROM EMPLOYEE E
JOIN PAYROLL_HIST PHST ON E.EMP_ID = PHST.EMP_ID
JOIN PAYROLL P ON PHST.PAY_ID = P.PAY_ID
WHERE E.EMP_FNAME = EMPLOYEE_FNAME AND E.EMP_LNAME = EMPLOYEE_LNAME
ORDER BY PAY_DATE
LIMIT 1;
```

```
CALL EmployeeData ("RAJ","KAPOOR");
```

Result Set: The result set returned includes the current pay position and the current pay amount of the employee. Note that the employee may previously be on a different position and be paid differently.

EMP_FNAME	EMP_LNAME	EMP_PHONE	PAY_POSITION	PAY_AMOUNT
RAJ	KAPOOR	9879879871	Electronic Press IRM	2000.00

II. VIEWS

1) The view CustomerOrderDetailData gives out information about the customer who ordered a product, the required date for order completion, the description of the product ordered, the quantity ordered and the information on which and how many pieces of a raw material are required for stitching that apparel. Now all this information is essential for the store manager to know in order to instruct tailors for stitching, handing over raw materials and getting orders completed on time. The full access to raw material table is restricted through this, in order to hide the raw material price. This is essential from the owner's view point as access to the billing data (both price and vendor) is something the store manager should not access. The customer_id is provided as it helps to view customer's size from the measurement table.

```
CREATE VIEW CustomerOrderDetailData AS
SELECT C.CUST_ID, C.CUST_FNAME, C.CUST_LNAME, O.ORDER_REQUIRED_DATE,
M.MANU_PRODUCT_DESC,
M.MANU_PRODUCT_QUANTITY, RA.RAW_MAT_NAME, R.REQ_QUANTITY_USED
FROM CUSTOMER C
JOIN `ORDER` O ON C.CUST_ID = O.CUST_ID
JOIN MANUFACTURED_PRODUCT M ON M.ORDER_ID = O.ORDER_ID
JOIN `REQUIRE` R ON R.MANU_PRODUCT_ID = M.MANU_PRODUCT_ID
JOIN RAW_MATERIAL  RA ON RA.RAW_MAT_ID = R.RAW_MAT_ID ;
```

2) The EmployeeEntry view is essential for the store manager in order to know the employee details. For example, which tailor is a Designer tailor or which employee works on a contract-basis and who is a full-time employee. Similar attributes need to be noted for ironing man as well while assigning work. Essentially, it helps to streamline the process of assigning hours of work to different employees according to their skill and position and ensuring the orders are completed on time for the customers. The employee details such as his/her address and hire date are hidden as the store manager need not see the personal details of an employee except the phone number.

```
CREATE VIEW EmployeeEntry AS

SELECT E.EMP_ID, E.EMP_FNAME, E.EMP_LNAME, E.EMP_PHONE,
E.EMP_TYPE, T.TAILOR_TYPE, T.TAILOR_ON_CONTRACT, IM.CAN_DRY_CLEAN, IM.PRESS_TYPE,
SM.EDUCATION_LEVEL, SM.CAN_OPERATE_LAPTOP
FROM EMPLOYEE E
LEFT JOIN TAILOR T ON E.EMP_ID = T.EMP_ID
LEFT JOIN IRONING_MAN IM ON E.EMP_ID = IM.EMP_ID
LEFT JOIN STORE_MANAGER SM ON E.EMP_ID = SM.EMP_ID
```