|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Creating a Database & performing SQL queries  SQL |
| |  |  |  | | --- | --- | --- | | Teresa Ventaja | 31-03-2020 | [Course title] | |

INDEX

[Background and Description 2](#_Toc37858449)

[ERD Diagram 2](#_Toc37858450)

[Resolution to exercises 4](#_Toc37858451)

[1. Create a View showing all transactions for a given week in your business. 4](#_Toc37858452)

[Graph 1 5](#_Toc37858453)

[2. Create a trigger that stores stock levels once a sale takes place. 6](#_Toc37858454)

[3. Create a View of stock (by supplier) purchased by you. 9](#_Toc37858455)

[4. Create a View of Total stock sold to general public (group by supplier). 10](#_Toc37858456)

[5. Detail and total all transactions (SALES) for the month-to-date. (*A Group By with Roll-Up*) 11](#_Toc37858457)

[6. Detail and total all SALES for the year-to-date. (*A Group By with Roll-Up*) 12](#_Toc37858458)

[Graph 2 13](#_Toc37858459)

[7. Detail & total transactions broken down on a monthly basis for 1 year. (*A Group By with Roll-Up*) 14](#_Toc37858460)

[8. Display the growth in sales/services (as a percentage) for your business, from the 1st month of opening until now. 15](#_Toc37858461)

# Background and Description

I have created a database for a store that sells drones. There are multiple choices of products covering the 4 different types of drones that we can find in the market. The database is prepared to process payments and store credit card information. It contains tables for both orders and returns. I have created sample data from the internet to load it into my database, so I can test the code.

I am adding a Script document where all the code is compiled and commented. On this report, I am going to include the ERD Diagram and the resolution to the CA questions, along with the 2 graphs required and screenshots of the code output.

# ERD Diagram

A close up of a map

Description automatically generated

# Resolution to exercises

## 1. Create a View showing all transactions for a given week in your business.

# Creating DataMart

Create Database DataMart;

# Creating view for Exercise 1

Use drone\_store;

Create VIEW Exercise1 AS

SELECT Orders.OrdersDate, payments.TransactionID, orders.OrderID, payments.FinalPrice, payments.CardNumber, payments.PaymentStatus

FROM Orders

INNER JOIN payments

ON orders.OrderID = payments.OrderID

WHERE orders.OrdersDate

HAVING OrdersDate BETWEEN '2019-08-12' AND '2019-08-18';

# Creating a table in the DataMart with the view

# Inserting the data created in the view (drone\_store) to my DataMart

USE DataMart;

Create Table DataMart.Exercise1\_Data SELECT \* FROM drone\_store.exercise1;

# Resolution to Exercise 1

USE DataMart;

SELECT \* FROM exercise1\_data;

A screenshot of a social media post

Description automatically generated

### Graph 1

The following chart, generated based on the data gathered in the View, shows the total transactions by payment status for week 12/08/2019 to 18/08/2019, classified by day:

## 2. Create a trigger that stores stock levels once a sale takes place.

# Creating audit table for exercise 2

USE drone\_store;

create table stock\_audit(

stockID int,

itemtitle varchar(100),

costprice varchar(40),

stocklevel int,

supplierID int

);

# Creating trigger for auditing all changes (exercise 2)

DELIMITER $$

CREATE TRIGGER before\_update\_stock

BEFORE UPDATE ON stock

FOR EACH ROW BEGIN

INSERT INTO stock\_audit

SET

StockID = OLD.stockID,

itemtitle = OLD.itemtitle,

costprice = old.costprice,

stocklevel = old.stocklevel,

supplierID = old.supplierID;

END$$

DELIMITER ;

# Creating a new trigger to update stock levels once a sale takes place

DELIMITER $$

CREATE TRIGGER update\_stock

AFTER INSERT ON orderitems

FOR EACH ROW

BEGIN

UPDATE stock

SET

stocklevel = stocklevel - new.quantitysold

where StockID = new.stockID;

END$$

DELIMITER ;

# Inserting a new row into the sales

USE DRONE\_STORE;

INSERT INTO orderitems (BarCode, StockID, item, QuantitySold, SalePrice, VAT, OrderID)

VALUES ('52667-2423', '12', 'Multi Rotor Drone', '1', '€1257,46', '€0,00', '121212');

#Resolution of Exercise 2

# Checking if Stock Level changed

SELECT \* FROM drone\_store.stock;

# Checking if changes were recorded in the audit table

SELECT \* FROM drone\_store.stock\_audit;

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

## 3. Create a View of stock (by supplier) purchased by you.

# Creating view for Exercise 3

Use drone\_store;

Create VIEW Exercise3 AS

SELECT suppliers.SupplierID, suppliers.SupName, stock.ItemTitle, stock.CostPrice, stock.StockLevel

FROM suppliers

INNER JOIN stock

ON suppliers.SupplierID = stock.SupplierID

ORDER BY SupplierID DESC;

# Creating a table in the DataMart with the view

# Inserting the data created in the view (drone\_store) to my DataMart

USE DataMart;

Create Table DataMart.Exercise3\_Data SELECT \* FROM drone\_store.exercise3;

# Resolution to Exercise 3

USE DataMart;

SELECT \* FROM exercise3\_data;

A screenshot of a computer

Description automatically generated

## 4. Create a View of Total stock sold to general public (group by supplier).

# Creating view for Exercise 4

# Using right join because there are more orders than stock and suppliers

Use drone\_store;

Create VIEW Exercise4 AS

SELECT stock.supplierID, stock.stockID, orders.orderid, orderitems.quantitysold, orderitems.saleprice

FROM stock

RIGHT JOIN orderitems

ON stock.stockID = orderitems.orderid

right join orders

ON orderitems.orderid = orders.orderid

order BY stock.SupplierID;

# Creating a table in the DataMart with the view

# Inserting the data created in the view (drone\_store) to my DataMart

USE DataMart;

Create Table DataMart.Exercise4\_Data SELECT \* FROM drone\_store.exercise4;

# Resolution to Exercise 4

USE DataMart;

SELECT \* FROM exercise4\_data

order BY supplierID;

A screenshot of a computer

Description automatically generated

## 5. Detail and total all transactions (SALES) for the month-to-date. (*A Group By with Roll-Up*)

# Resolution to exercise 5

# Using SUBSTRING because my prices have the symbol €

USE drone\_store;

SELECT coalesce (orderitems.item,'GrandTotal') AS Salesbyitem,

Sum(SUBSTRING(orderitems.SalePrice, 2)) as SummorizedPurchaseAmt

FROM orderitems

INNER JOIN orders

ON orderitems.orderID = orders.orderid

WHERE orders.ordersdate BETWEEN '2020-03-01' AND '2020-03-31'

GROUP BY Salesbyitem WITH ROLLUP;

A screenshot of a social media post

Description automatically generated

## 6. Detail and total all SALES for the year-to-date. (*A Group By with Roll-Up*)

# Resolution to exercise 6

# Using SUBSTRING because my prices have the symbol €

USE drone\_store;

SELECT coalesce (orderitems.item,'GrandTotal') AS Salesbyitem2020,

Sum(SUBSTRING(orderitems.SalePrice, 2)) as TotalSales

FROM orderitems

INNER JOIN orders

ON orderitems.orderID = orders.orderid

WHERE orders.ordersdate BETWEEN '2020-01-01' AND '2020-03-31'

GROUP BY Salesbyitem2020 WITH ROLLUP;

A screenshot of a computer

Description automatically generated

### Graph 2

The following pie chart shows details on the total sales for 2020, a representation of the percentage of product sold up to now, 31st of March 2020:

## 7. Detail & total transactions broken down on a monthly basis for 1 year. (*A Group By with Roll-Up*)

# Resolution to exercise 7

# Using SUBSTRING because my prices have the symbol €

USE drone\_store;

SELECT MONTHNAME(orders.ordersdate) AS '2019',

count(distinct(orders.orderid)) as NumberofOrders,

Sum(SUBSTRING(orderitems.SalePrice, 2)) as TotalSales

FROM orderitems

INNER JOIN orders

ON orderitems.orderID = orders.orderid

WHERE orders.ordersdate BETWEEN '2019-01-01' AND '2019-12-31'

GROUP BY MONTH(orders.ordersdate) WITH ROLLUP

ORDER BY orders.ordersdate;

A screenshot of a computer

Description automatically generated

## 8. Display the growth in sales/services (as a percentage) for your business, from the 1st month of opening until now.

# Resolution to exercise 8

# Using SUBSTRING because my prices have the symbol €

USE drone\_store;

SELECT YEAR(orders.ordersdate) AS 'Year',

MONTHNAME(orders.ordersdate) AS 'Month',

count(distinct(orders.orderid)) as NumberofOrders,

Sum(SUBSTRING(orderitems.SalePrice, 2)) as TotalSales,

((sum((SUBSTRING(orderitems.SalePrice, 2)) / 1429204)\*100)) as percentage

FROM orderitems

INNER JOIN orders

ON orderitems.orderID = orders.orderid

WHERE orders.ordersdate BETWEEN '2019-01-01' AND '2020-03-31'

GROUP BY MONTH(orders.ordersdate) WITH ROLLUP

ORDER BY orders.ordersdate;

A screenshot of a computer

Description automatically generated