2814ICT – DATA MANAGEMENT
7003ICT – DATABASE DESIGN
School of Information & Communication Technology
Trimester 1, 2021

Assignment Part 2:
Implementing a Database for BigM

ASSIGNMENT TITLE: Analysing ERDs and Writing corresponding SQL statements for BigM's Database.

Group ID	**011**	
Student 1	**s-number: s5196431**	**Full name: Duwon Ha**
Student 2	**s-number: s5203114**	**Full name: Kavya Krishnakumar**
Student 3	**s-number: s5213262**	**Full name: Shinzo Tanimoto**
Course Code: 2814ICT		**Workshop/Lab day & time: Friday 9:00am – 10:45am**
Tutor's name: Emon Kumar Dey		**Date submitted: 25/05/2021**

Contents

List of Illustrations

Acknowledgements

1) Emon Kumar Dey

Task 1: Creating SQL Tables

```
CREATE DATABASE IF NOT EXISTS BigM_s5213262;

USE bigm_s5213262;

SHOW TABLES;

CREATE TABLE IF NOT EXISTS CUSTOMER(
	Cust_Number		INT PRIMARY KEY AUTO_INCREMENT,
	Cust_Fname		VARCHAR(30),
	Cust_Lname			VARCHAR(30),
	Cust_Phone			CHAR(10)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS PRODUCT(
	Prod_Num		INT PRIMARY KEY AUTO_INCREMENT,
	Prod_Desc		VARCHAR(30),
	Prod_Size		VARCHAR(30),
	Prod_Price		DECIMAL
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS DEPARTMENT(
	Dept_ID		INT PRIMARY KEY AUTO_INCREMENT,
	Dept_Name		VARCHAR(30)
) ENGINE=InnoDB;

CREATE TABLE  IF NOT EXISTS EMPLOYEE(
	Emp_ID		INT PRIMARY KEY AUTO_INCREMENT,
	Emp_FName	VARCHAR(30),
	Emp_LName	VARCHAR(30),
	Emp_Phone	INTEGER,
	Emp_DoB		DATE,
	Emp_StartDate		DATE,
	Emp_TaxFNum		INTEGER,
	Emp_HourlySalary	DECIMAL(10,2),
	StrDept_ID		INT,
	SupervisorID	INT
) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS STORE(
	Str_Num		INT PRIMARY KEY AUTO_INCREMENT,
	Str_Name		VARCHAR(50),
	Str_Phone		CHAR(10),
	Str_Fax		CHAR(10),
	Str_Email		VARCHAR(40),
	StoreManagerID		INT,
	SupStore_Num		INT,
	FOREIGN KEY (StoreManagerID)	REFERENCES EMPLOYEE(Emp_ID),
```

```
	FOREIGN KEY (SupStore_Num)	REFERENCES STORE(Str_Num)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS CUSTOMERORDER(
	CustOrd_ID				INT PRIMARY KEY AUTO_INCREMENT,
	CustOrd_Date		DATE,
	Cust_Number				INT NOT NULL,
	Str_Num				INT NOT NULL,
	FOREIGN KEY (Cust_Number)	REFERENCES CUSTOMER (Cust_Number),
	FOREIGN KEY (Str_Num)		REFERENCES STORE (Str_Num)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS ORDERLINE(
	CustOrd_ID		INT,
	Prod_Num		INT,
	OrdLn_DateArrived	DATE,
	OrdLn_DatePicked	DATE,
	OrdLn_Qnty		INT,
	PRIMARY KEY (CustOrd_ID, Prod_Num),
	FOREIGN KEY (CustOrd_ID) REFERENCES CUSTOMERORDER(CustOrd_ID),
	FOREIGN KEY (Prod_Num) REFERENCES PRODUCT (Prod_Num)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS STOREDEPARTMENT(
	StrDept_ID				INT PRIMARY KEY AUTO_INCREMENT,
	StrDept_Phone		CHAR(10),
	StrDept_Email		VARCHAR(40),
	DeptSupervisorID		INT,
	Str_Num				INT,
	Dept_ID				INT,
	FOREIGN KEY (DeptSupervisorID)	REFERENCES EMPLOYEE(Emp_ID),
	FOREIGN KEY (Str_Num)	REFERENCESSTORE(Str_Num),
	FOREIGN KEY (Dept_ID)	REFERENCESDEPARTMENT(Dept_ID)
) ENGINE=InnoDB;

ALTER TABLE EMPLOYEE
ADD FOREIGN KEY (StrDept_ID) REFERENCES STOREDEPARTMENT (StrDept_ID);
ALTER TABLE EMPLOYEE
ADD FOREIGN KEY (SupervisorID) REFERENCES EMPLOYEE (Emp_ID);

CREATE TABLE IF NOT EXISTS PAYSLIP(
	Pay_ID		INT PRIMARY KEY AUTO_INCREMENT,
	Pay_date		DATE,
	Pay_num_of_hours		INT NOT NULL,
	Pay_amount_gross		DOUBLE NOT NULL,
	Emp_ID		INT NOT NULL,
	Str_Num		INT NOT NULL,
	FOREIGN KEY (Emp_ID)		REFERENCES EMPLOYEE (Emp_ID),
	FOREIGN KEY (Str_Num)		REFERENCES STORE (Str_Num)
) ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS INVENTORY(
	ProductNum		INT,
	Str_Num		INT,
	Inv_QntyOnHand		INT NOT NULL,
	Inv_QtyOrdered		INT NOT NULL,
	PRIMARY KEY (ProductNum, Str_Num),
	FOREIGN KEY (ProductNum) REFERENCES PRODUCT (Prod_Num),
	FOREIGN KEY (Str_Num)		REFERENCES STORE (Str_Num)
) ENGINE=InnoDB;
```

Task 2: Inserting Records

```
USE bigm_s5213262;

INSERT INTO CUSTOMER VALUES
	(NULL, 'Buffy', 'Winters', '0412345678'),
	(NULL, 'Duwon', 'Ha', '0455803205'),
	(NULL, 'Kavya', 'Krishnakumar', '0489789774'),
	(NULL, 'Jack', 'Tomlinson', '0484787854'),
	(NULL, 'Shinzo', 'Tanimoto', '0435607767'),
	(NULL, 'Jone', 'Tyler', '0445679873'),
	(NULL, 'Princess', 'Ha', '0413247477'),
	(NULL, 'David', 'Bieber', '0412522377');


INSERT INTO PRODUCT VALUES
	(NULL, 'Gardening Scissors','Medium', 10),
	(NULL, 'Shirt', 'Medium', 30),
	(NULL, 'Kitchen Knives Set Deluxe', 'Medium', 120),
	(NULL, 'Basketball Regulation Size', 'Small', 350),
	(NULL, 'Nickson James Hits', 'Small', 50),
	(NULL, '100% Polyester sleepwear', 'Large', 120),
	(NULL, 'Jordan Shoes', 'Small', 700);

INSERT INTO DEPARTMENT VALUES
	(NULL, 'Sports'),
	(NULL, 'Clothes'),
	(NULL, 'Kitchen'),
	(NULL, 'Homeware'),
	(NULL, 'Garden');

INSERT INTO EMPLOYEE(Emp_ID, Emp_FName, Emp_LName, Emp_Phone, Emp_DoB,
Emp_StartDate, Emp_TaxFNum, Emp_HourlySalary) VALUES
	(NULL, 'Koupa', 'Taylor', '0321224224', '2002-03-12', '2020-05-02', 12345678, 30),
	(NULL, 'Ben', 'Flint', '0434551234', '1999-06-12', '2020-05-02', 15125677, 60),
	(NULL, 'Harry', 'Robin', '0449879343', '2002-07-07', '2020-09-12', 98765432, 30),
	(NULL, 'Elizabeth', 'Then', '0421252322', '1998-03-23', '2021-03-02', 9823124, 30),
	(NULL, 'Jordan', 'Nickson', '0414279846', '1989-08-01', '2010-04-20', 09872525, 40),
	(NULL, 'Jason', 'Micheal', '0412340987', '1997-05-04', '2019-12-12', 102938475, 30);

INSERT INTO STORE(Str_Num, Str_Name, Str_Phone, Str_Fax, Str_Email,
StoreManagerID) VALUES
	(NULL, 'Elizabeth-Street-Branch', '0312452524', '0145252434', 'elizabeth@bigm.au',
3),
	(NULL, 'Adelaide-Street-Branch', '0124352321', '291232423', 'adelaide@bigm.au', 1),
	(NULL, 'Southbank-Branch', '023674845', '026882683', 'southbank@bigm.au', 4),
	(NULL, 'Surfers-Paradise-Branch', '0123452352', '2948274817',
'surfers.paradise@bigm.au', 5),
	(NULL, 'Mount-Gravatt-Branch', '0937192322', '2982142351',
'mountGravatt@bigm.au', 6);
```

```
INSERT INTO CUSTOMERORDER VALUES
(NULL, '2021-05-03', 1, 1),
(NULL, '2021-05-07', 2, 2),
(NULL, '2021-05-22', 3, 3),
(NULL, '2021-05-12', 4, 4),
(NULL, '2021-05-25', 5, 5),
(NULL, '2021-05-30', 6, 3),
(NULL, '2021-05-22', 7, 4);

INSERT INTO ORDERLINE VALUES
	(1, 1, '2017-06-03', '2017-06-04', 3),
	(2, 2, '2021-06-07', '2021-06-09', 2),
	(3, 3, '2015-06-22', '2015-06-23', 2),
	(4, 4, '2021-06-12', '2021-06-13', 1),
	(5, 5, '2014-06-17', '2014-06-20', 3);

INSERT INTO STOREDEPARTMENT VALUES
	(NULL, '1234567891', 'elizabeth.kitchen@bigm.au', 3, 1, 3),
	(NULL, '1209384756', 'adelaide.sports@bigm.au', 5, 2, 1),
	(NULL, '1252156215', 'southbank.kitchen@bigm.au', 4, 3, 3),
	(NULL, '9830484921', 'surfers.paradice.homewear@bigm.au', 1, 4, 4),
	(NULL, '8273182931', 'mountGravatt.garden@bigm.au', 6, 5, 5),
	(NULL, '2910242312', 'elizabeth.garden@bigm.au', 3, 1, 5),
	(NULL, '2923144212', 'adelaide.garden@bigm.au', 2, 2, 5);

UPDATE EMPLOYEE SET StrDept_ID = 4 WHERE Emp_ID = 1;
UPDATE EMPLOYEE SET StrDept_ID = 7 WHERE Emp_ID = 2;
UPDATE EMPLOYEE SET StrDept_ID = 1 WHERE Emp_ID = 3;
UPDATE EMPLOYEE SET StrDept_ID = 3 WHERE Emp_ID = 4;
UPDATE EMPLOYEE SET StrDept_ID = 2 WHERE Emp_ID = 5;
UPDATE EMPLOYEE SET StrDept_ID = 5 WHERE Emp_ID = 6;

UPDATE STORE SET SupStore_Num = 2 WHERE Str_Num = 1;
UPDATE STORE SET SupStore_Num = 3 WHERE Str_Num = 2;
UPDATE STORE SET SupStore_Num = 4 WHERE Str_Num = 3;
UPDATE STORE SET SupStore_Num = 5 WHERE Str_Num = 4;
UPDATE STORE SET SupStore_Num = 1 WHERE Str_Num = 5;

INSERT INTO PAYSLIP VALUES
	(NULL, '2021-06-30', 80, 2400 ,1,2),
	(NULL, '2021-06-30', 72, 4320, 2,2),
	(NULL, '2021-06-30', 50, 1500, 3,1),
	(NULL, '2021-06-30', 60, 1800, 4,3),
	(NULL, '2021-06-30', 80, 3200, 5,4),
	(NULL, '2021-06-30', 68, 2040, 6,5);

INSERT INTO INVENTORY VALUES
	(2,2,3,4),
	(1,3,4,5),
	(3,2,1,10),
	(2,3,10,2),
	(5,4,2,4),
	(5,3,2,2),
	(6,5,5,2),
	(7,2,5,2);
```

Task 3: SQL Queries

Query 1: List of names of all employees sorted by their hourly salary.

```
SELECT CONCAT(Emp_FName, ' ', Emp_LName) AS 'Name'
FROM EMPLOYEE
ORDER BY Emp_HourlySalary;
```

Output table:

```
SELECT CONCAT(Emp_FName, ' ', Emp_LName) AS NAME FROM EMPLOYEE ORDER BY Emp_HourlySalary
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

NAME
Koupa Taylor
Harry Robin
Elizabeth Then
Jason Micheal
Jordan Nickson
Ben Flint

Table 1

Query 2: The date on which the most recent customer order has been made. The customer's name and date of the order will be sufficient.

```
SELECT CONCAT(C.Cust_FName, ' ' , C.Cust_LName) AS 'Name', O.CustOrd_Date AS
'Date Of The Order'
FROM CUSTOMER AS C, CUSTOMERORDER AS O
WHERE C.Cust_Number = O.Cust_Number
Order by CustOrd_Date DESC
limit 0,1;
```

Output table:

Showing rows 0 - 0 (1 total, Query took 0.0022 seconds.)

```
SELECT CONCAT(C.Cust_FName, ' ' , C.Cust_LName) AS 'Name', O.CustOrd_Date AS 'Date Of The Order' FROM CUSTOMER AS C, CUSTOMERORDER AS O WHERE C.Cust_Number = O.Cust_Number Order by
CustOrd_Date DESC limit 0,1
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

	Name	Date Of The Order
Edit Copy Delete	Jone Tyler	2021-05-30

Check all With selected: Edit Copy Delete Export

Table 2

Query 3: List of all the store names and their manager names, sorted in dictionary order of the store name.

```
SELECT S.Str_Name AS 'Store Name', CONCAT(E.Emp_FName, ' ', E.Emp_LName) AS
'Name'
FROM STORE AS S, EMPLOYEE AS E
WHERE S.StoreManagerID = E.Emp_ID
ORDER BY S.Str_Name ASC;
```

Output table:

Showing rows 0 - 4 (5 total, Query took 0.0027 seconds.)

```
SELECT S.Str_Name AS 'Store Name', CONCAT(E.Emp_FName, ' ', E.Emp_LName) AS 'Name' FROM STORE AS S, EMPLOYEE AS E WHERE S.StoreManagerID = E.Emp_ID ORDER BY S.Str_Name ASC
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	Store Name	Name
Edit Copy Delete	Adelaide-Street-Branch	Koupa Taylor
Edit Copy Delete	Elizabeth-Street-Branch	Harry Robin
Edit Copy Delete	Mount-Gravatt-Branch	Jason Micheal
Edit Copy Delete	Southbank-Branch	Elizabeth Then
Edit Copy Delete	Surfers-Paradise-Branch	Jordan Nickson

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

Table 3

Query 4: A list of all customers that have not placed an order yet. Displaying customer number and name will be sufficient.

SELECT C.Cust_Number AS 'Custom Number', CONCAT(C.Cust_FName, ' ', C.Cust_LName) AS 'Name'
FROM CUSTOMER AS C
WHERE C.Cust_Number NOT IN (SELECT Cust_Number FROM CUSTOMERORDER);

Output table:

Showing rows 0 - 0 (1 total, Query took 0.0025 seconds.)

```
SELECT C.Cust_Number AS 'Custom Number', CONCAT(C.Cust_FName, ' ', C.Cust_LName) AS 'Name' FROM CUSTOMER AS C WHERE C.Cust_Number NOT IN (SELECT Cust_Number FROM CUSTOMERORDER)
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	Custom Number	Name
Edit Copy Delete	8	David Bieber

Check all With selected: Edit Copy Delete Export

Table 4

Query 5: A list containing the name of employees, who work as supervisors for 'Sports' departments in various stores. Show store names and the supervisors of Sports department.

SELECT S.Str_Name AS 'Store Name', CONCAT(E.Emp_FName, ' ', E.Emp_LName) AS 'Supervisor Name'
FROM STORE AS S, EMPLOYEE AS E, STOREDEPARTMENT AS SD
WHERE SD.DeptSupervisorID = E.Emp_ID
AND SD.Dept_ID = (SELECT Dept_ID FROM DEPARTMENT WHERE Dept_Name = 'Sports')
AND SD.Str_Num = S.Str_Num;

Output table:

Showing rows 0 - 0 (1 total, Query took 0.0054 seconds.)

```
SELECT S.Str_Name AS 'Store Name', CONCAT(E.Emp_FName, ' ', E.Emp_LName) AS 'Supervisor Name' FROM STORE AS S, EMPLOYEE AS E, STOREDEPARTMENT AS SD WHERE SD.DeptSupervisorID =
E.Emp_ID AND SD.Dept_ID = (SELECT Dept_ID FROM DEPARTMENT WHERE Dept_Name = 'Sports') AND SD.Str_Num = S.Str_Num
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	Store Name	Supervisor Name
Edit Copy Delete	Adelaide-Street-Branch	Jordan Nickson

Table 5

Query 6: A list containing the total quantity on hand for each product (product number and description) regardless of stores.

```
SELECT P.Prod_Num AS 'Product Number', P.Prod_Desc AS 'Description',
SUM(Inv_QntyOnHand) AS 'Quantity On Hand'
FROM PRODUCT AS P, INVENTORY AS I
WHERE P.Prod_Num = I.ProductNum
GROUP BY P.Prod_Num;
```

Output table:

Showing rows 0 - 5 (6 total, Query took 0.0024 seconds.)

```
SELECT P.Prod_Num AS 'Product Number', P.Prod_Desc AS 'Description', SUM(Inv_QntyOnHand) AS 'Quantity On Hand' FROM PRODUCT AS P, INVENTORY AS I WHERE P.Prod_Num = I.ProductNum
GROUP BY P.Prod_Num
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	Product Number	Description	Quantity On Hand
Edit Copy Delete	1	Gardening Scissors	4
Edit Copy Delete	2	Sweatshirt	13
Edit Copy Delete	3	Kitchen Knives Set Deluxe	1
Edit Copy Delete	5	Nickson James Hits	4
Edit Copy Delete	6	100% Polyester sleepwear	5
Edit Copy Delete	7	Jordan Shoes	5

Table 6

Query 7: A list showing each product sold (picked) on or before May 20, 2018. Show product number, name and quantity sold, sorted by product number and then quantity sold.

```
SELECT P.Prod_Num AS 'Product Number', P.Prod_Desc AS
'Description',  SUM(O.OrdLn_Qnty) AS 'Quantity Sold'
FROM ORDERLINE AS O, PRODUCT AS P
WHERE   O.Prod_Num = P.Prod_Num AND OrdLn_DatePicked <= '2018-05-20'
GROUP BY P.Prod_Num
ORDER BY P.Prod_Num, sum(O.OrdLn_Qnty);
```

Output table:

Showing rows 0 - 2 (3 total, Query took 0.0031 seconds.)

```
SELECT P.Prod_Num AS 'Product Number', P.Prod_Desc AS 'Description', SUM(O.OrdLn_Qnty) AS 'Quantity Sold' FROM ORDERLINE AS O, PRODUCT AS P WHERE O.Prod_Num = P.Prod_Num AND
OrdLn_DatePicked <= '2018-05-20' GROUP BY P.Prod_Num ORDER BY P.Prod_Num, sum(O.OrdLn_Qnty)
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	Product Number	Description	Quantity Sold
Edit Copy Delete	1	Gardening Scissors	3
Edit Copy Delete	3	Kitchen Knives Set Deluxe	2
Edit Copy Delete	5	Nickson James Hits	3

Check all *With selected:* Edit Copy Delete Export

Table 7

Query 8: A list of products (show product number, description and price) whose price is less than or equal to the average product price.

```
SELECT Prod_Num AS 'Product Number', Prod_Desc 'Description', Prod_Price AS '($) Price'
FROM PRODUCT
WHERE Prod_Price <= (SELECT AVG(Prod_Price) FROM PRODUCT);
```

Output table:

Showing rows 0 - 4 (5 total, Query took 0.0019 seconds.)

```
SELECT Prod_Num AS 'Product Number', Prod_Desc 'Description', Prod_Price AS '($) Price' FROM PRODUCT WHERE Prod_Price <= (SELECT AVG(Prod_Price) FROM PRODUCT)
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Product Number	Description	($) Price
Edit Copy Delete	1	Gardening Scissors	10
Edit Copy Delete	3	Kitchen Knives Set Deluxe	100
Edit Copy Delete	4	Basketball Regulation Size	350
Edit Copy Delete	5	Nickson James Hits	50
Edit Copy Delete	7	Jordan Shoes	400

Table 8

Query 9: Increase each employee's salary by 7.5% and show the updated salary of all employees (name and salary).

```
UPDATE EMPLOYEE
SET Emp_HourlySalary = Emp_HourlySalary*1.075;

SELECT CONCAT(Emp_FName, ' ', Emp_LName) AS 'Name', Emp_HourlySalary AS 'Hourly Salary'
FROM EMPLOYEE;
```

Output table:

Showing rows 0 - 5 (6 total, Query took 0.0013 seconds.)

```
SELECT CONCAT(Emp_FName, ' ', Emp_LName) AS 'Name', Emp_HourlySalary AS 'Hourly Salary' FROM EMPLOYEE
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Name	Hourly Salary
Edit Copy Delete	Koupa Taylor	32.25
Edit Copy Delete	Ben Flint	64.50
Edit Copy Delete	Harry Robin	32.25
Edit Copy Delete	Elizabeth Then	32.25
Edit Copy Delete	Jordan Nickson	43.00
Edit Copy Delete	Jason Micheal	32.25

Table 9

Query 10: Show the pay information (employee name, hours paid, amount paid) of all employees in the most recent pay date.

```
SELECT CONCAT(E.Emp_FName, ' ', E.Emp_LName) AS 'Name',
CAST((P.Pay_amount_gross / E.Emp_HourlySalary) AS INT) AS 'Hours Paid',
P.Pay_amount_gross AS 'Total Amount Paid'
FROM EMPLOYEE AS E, PAYSLIP AS P
WHERE P.Emp_ID = E.Emp_ID
AND P.Pay_date = (SELECT Pay_date
                  FROM PAYSLIP
                  ORDER BY Pay_date DESC
                  limit 0, 1);
```

Output table:

Showing rows 0 - 5 (6 total, Query took 0.0035 seconds.)

```
SELECT CONCAT(E.Emp_FName, ' ', E.Emp_LName) AS 'Name', CAST((P.Pay_amount_gross / E.Emp_HourlySalary) AS INT) AS 'Hours Paid', P.Pay_amount_gross AS 'Total Amount Paid' FROM
EMPLOYEE AS E, PAYSLIP AS P WHERE P.Emp_ID = E.Emp_ID AND P.Pay_date = (SELECT Pay_date FROM PAYSLIP ORDER BY Pay_date DESC limit 0, 1)
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	Name	Hours Paid	Total Amount Paid
Edit Copy Delete	Koupa Taylor	74	2400
Edit Copy Delete	Ben Flint	67	4320
Edit Copy Delete	Harry Robin	47	1500
Edit Copy Delete	Elizabeth Then	56	1800
Edit Copy Delete	Jordan Nickson	74	3200
Edit Copy Delete	Jason Micheal	63	2040

Table 10

Task 4: Inserting Additional Data

```
USE bigm_s5213262;

INSERT INTO CUSTOMER VALUES
(NULL, 'Daniel', 'Ortega', '0431xxx668');

INSERT INTO CUSTOMERORDER VALUES
(NULL, '2018-09-06', (SELECT Cust_Number FROM CUSTOMER ORDER BY Cust_Number
DESC limit 0, 1), 2);

INSERT INTO ORDERLINE VALUES
((SELECT CustOrd_ID FROM CUSTOMERORDER ORDER BY CustOrd_ID DESC limit 0, 1),
2, '2018-09-08', '2018-09-10', 2);

UPDATE INVENTORY
SET Inv_QtyOnHand = Inv_QtyOnHand - (SELECT OrdLn_Qnty FROM ORDERLINE
WHERE CustOrd_ID = (SELECT CustOrd_ID FROM CUSTOMERORDER ORDER BY
CustOrd_ID DESC limit 0, 1) AND Prod_Num = 2)
WHERE ProductNum = 2
AND Str_Num = 2;

UPDATE INVENTORY
SET Inv_QtyOrdered = Inv_QtyOrdered + (SELECT OrdLn_Qnty FROM ORDERLINE
WHERE CustOrd_ID = (SELECT CustOrd_ID FROM CUSTOMERORDER ORDER BY
CustOrd_ID DESC limit 0, 1) AND Prod_Num = 2)
WHERE ProductNum = 2
AND Str_Num = 2;
```