COS20015 – Fundamentals of Data Management

Learning Summary Report

Masrur Rahman Zahin ( 101214608 )

**Graphical user interface, text

Description automatically generated**

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Pass (D) | Credit (C) | Distinction (B) | High Distinction (A) |
| Self-Assessment (please tick) |  |  | ✓ |  |

*Self-assessment Statement*

|  |  |
| --- | --- |
|  | Included (please tick) |
| Learning Summary Report | ✓ |
| Test is Complete | ✓ |
| Task reports that demonstrate coverage of core concepts | ✓ |

*Minimum Pass Checklist*

|  |  |
| --- | --- |
|  | Included (please tick) |
| Progress on Credit Tasks | ✓ |
| All Pass Tasks signed off | ✓ |

*Minimum Credit Checklist, in addition to Pass Checklist*

|  |  |
| --- | --- |
|  | Included (please tick) |
| Credit and Pass Tasks done, and Progress on Distinction Tasks. | ✓ |
| Custom data store meets Distinction criteria | ✓ |
| Design report with suitable illustrations of data store | ✓ |

*Minimum Distinction Checklist, in addition to Credit Checklist*

|  |  |
| --- | --- |
|  | Included (please tick) |
| Research report, and associated pieces | - |
| Custom project meets HD requirements | - |

*Minimum High Distinction Checklist, in addition to Distinction Checklist*

# Declaration

I declare that this portfolio is my individual work. I have not copied from any other student’s work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.



# Introduction

This report summarises what I have learnt in COS20015 Fundamentals of Data Management. It includes a self-assessment against the criteria described in the unit outline, a justification of the pieces included details of the coverage of the unit’s intended learning outcomes, and a reflection on my learning.

# Overview of Pieces Included

This section outlines the pieces that I have included in my portfolio…

* **Test**: The semester test I have attempted.
* **Text processing and Regular Expressions**: task about text processing and regular expression to match unstructured data.
* **Semi-Structured Data**: Task about XML and JSON on how to use Xquery and JSONiq to go through data.
* **Entity Relationship Modelling**: Task on how to properly draw an ERD diagram.
* **Normalisation**: Task on normalizing tables into 1NF, 2NF and 3NF.
* **Industry Case Study ERD**: Task on how to design a small database on an existing business
* **SQL - DML Queries**: Task on how to query a database using Select statement.
* **SQL - DML Updates**: Task on how to update a table in a database using Update statement.
* **Transactions and Concurrency**: Task on Transactions and Concurrency in a database.
* **Performance**: Task on optimization and how to improve performance while handling a database.
* **MongoDB**: task on noSQL database using MongoDB.
* **Distinction Report**: My distinction report that required me to create 10 entity and enter a minimum 10 entries per entity.

# Coverage of the Intended Learning Outcomes

This section outlines how the pieces I have included demonstrate the depth of my understanding in relation to each of the unit’s intended learning outcomes.

## ILO 1: Data Management Theory

The following pieces demonstrate my ability in relation to this ILO:

Task 7:

In this task we learnt about set theory and relational algebra, also we learnt how query works using operations such as selection, projection, cartesian product, unions, interactions and set difference to output the results. We also learnt about join statements and how DBMS uses these combinations to carry out the queries.

Task 10:

In this task we learnt about performance and some algorithmic complexity and how certain search can take longer as it uses the Big O(n) notation, the same search can be made faster if we sort our data first which then uses Big O(log n) search.

## ILO 2: Storing Data

Use techniques, tools and methods to sort, search and transform data stored in a variety of data formats.

Task 3:

In this task we learnt to store data in XML and JSON format which are semi-structured model. These formats are commonly used as they are human readable, they can also be parsed by other programming languages when exported. For this task I learnt how to use XPath and XQuery to work with XML file and JSONiq to work with JSON file.

Task 7:

In this task we learnt to return output from a database using the SELECT statement, based on how we write the select statement we can retrieve specific information from a table or even do calculations.

Task 8:

In this task we learnt how to insert or save data into tables using the INSERT statement, and how we can chose which row to insert into using the where clause.

Task 11:

In this task we learnt the same concept as the previous tasks but in MongoDB which is a NoSQL database, it had a slight different syntax for its queries but did complete the same operations as traditional SQL statements.

## ILO 3: Data Representation and Access

Explain the role of data types, data representation, indexing and schemas in managing data, and use methods to validate that data matches an expected schema.

Task 7 & 8:

In this task we learnt that during the creation of a table every column that goes inside the table just be attached with a data type to establish a consistent schema or a database design. So that wrong data cannot be inserted into a column that might cause errors later

We also did indexing to speed up the look up performance as well as to monitor the type of data being entered into the tables.

## ILO 4: Data Retrieval and Performance

Use appropriate methods to efficiently store, insert and retrieve data appreciating the underlying trade offs between different strategies.

Task 2, 3, 4, 7 ,8 & 10:

Throughout these tasks we learnt about the different types of data, the 3 different types of data are Unstructured, Semi-Structured and Structured data and they have their own advantages and disadvantages.

Unstructured data is one of the most common data type for example log files video files or audio file. Basically, any type of data that does not fit in a spreadsheet with rows and column. So, they cannot be parsed in a DBMS and must be pre-processed first

Semi-structured data can be parsed with a programming language, but it does not have a consistent structure or schema, thus it may need special error checking when it comes to reading data from semi-structured data such as XML or JSON files.

Structured data is the most common and useful for processing and analysis data type and it is easily searchable using simple algorithms. Compared to semi-structured data type we can make the data extremely useful without processing it much and querying it with SQL only. One of the main advantages of using Structured data is it’s schema consistency.

## ILO 5: Concurrency Considerations

Appreciate issues related to concurrency in data management and describe basic strategies for addressing these issues.

Task 9:

In this task we learnt about concurrency and how we can avoid issues that arise from multiple concurrent connections with the database. To avoid such inconsistent data or unintended updates we can turn auto-commit off and manually commit all successful queries into our database.

We can also implement different types of read isolation levels such as repeatable read, read committed or serializable isolations to minimize database anomalies.

# Reflection

## The most important things I learnt:

Coming to this unit I had no idea on how to manage a database, but by the end of the 12th week I think I have a good understanding on how to create and manage a database. Although there were a lot of ups and downs during my semester but the most important thing I learnt besides database was how to manage time and learn a new concept within it. We had new tasks every week and trying to complete them in time was sometimes a hassle by in the end I managed to do them all.

## The things that helped me most were:

* Educational websites such as W3School that helped me understand the topics better
* Communicating with lecturers regarding doubts on a particular topic or question.

## I found the following topics particularly challenging:

* I found regex kind of difficult since it was something very new I had to learn
* I did struggle trying to understand the cardinality and ERD diagrams at the start, I used to get stuck in this loop of thought that used to give me wrong results.

## I found the following topics particularly interesting:

* The creation of tables and insertion of data inside an entity using MySQL was very interesting and satisfying to me when all rows worked perfectly at once with no error.
* How the tables join and work together to give results was also very interesting to me

## I feel I learnt these topics, concepts, and/or tools really well:

* I learnt how to use SQL pretty well or at least the topics covered in my semester
* Got comfortable at using Draw.io and MySQL Workbench because of my tasks

## I still need to work on the following areas:

* I still sometimes struggle with cardinality and have to improve upon it more.

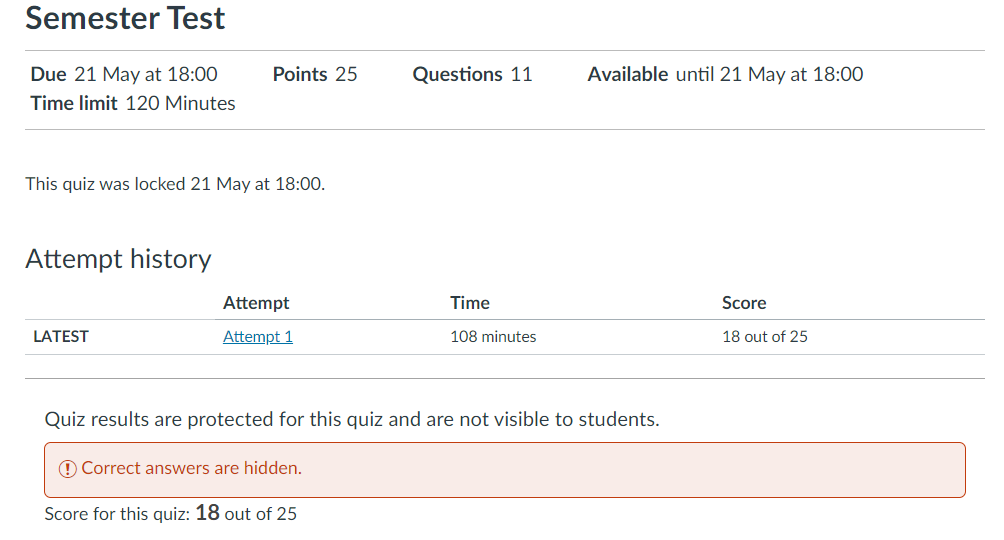
## This unit will help me in the future:

Not only I think this would help me become or at least open up a career path for me as a Data Manager, but it did help me in one of my unit (COS20019 Cloud Computing) where I had to manage a database, I did carry my knowledge from FDM to CC and it helped me immensely and made my workflow smoother.

## If I did this unit again, I would do the following things differently:

If I did this unit again, I assume I will have my knowledge of handling a database so in that case I would focus on time management and try to complete my tasks as soon as possible so I can focus on other units after that.

# Semester Test



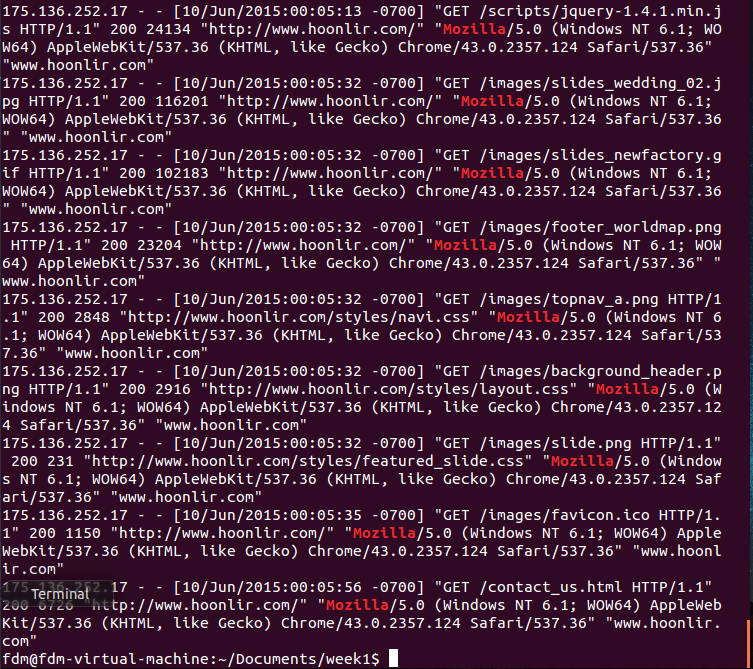
# **Tasks**

Pass Task 1.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 1.1.1

#### Solution

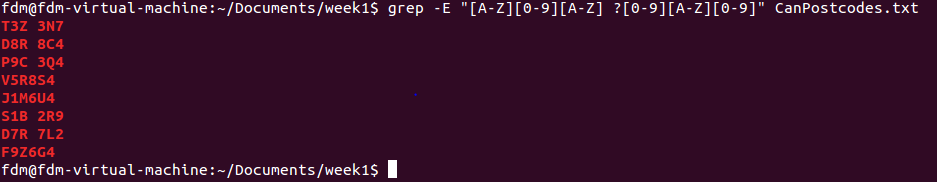


#### Comments/findings

This was not hard to do, it was pretty straight forward with the instructions.

## SUB Task 1.1.2

#### Solution



#### Comments/findings

I was having a hard time getting a result until I did some google search and found out I needed “ ” around my regex.

## SUB Task 1.1.3

#### Solution



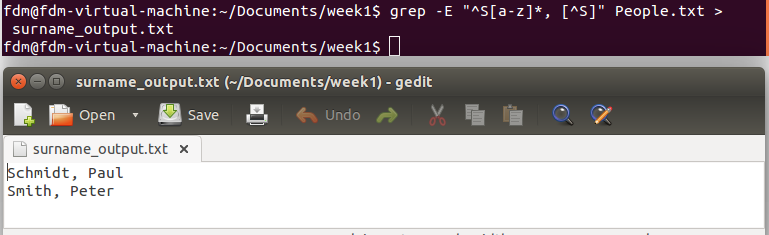
#### Comments/findings

This was not very hard once I found out that “^” symbol has two meaning, start with and not.

Then it was just a matter of telling what not to find.

## SUB Task 1.1.4

#### Solution

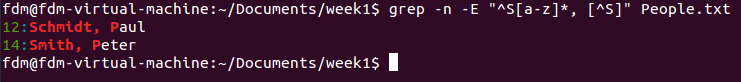


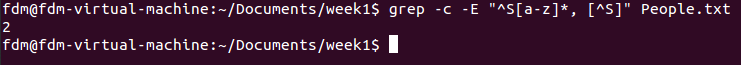
#### Comments/findings

From a quick google search I found out that I only needed “>” symbol to pipe data into a file.

## SUB Task 1.1.5

#### Solution





#### Comments/findings

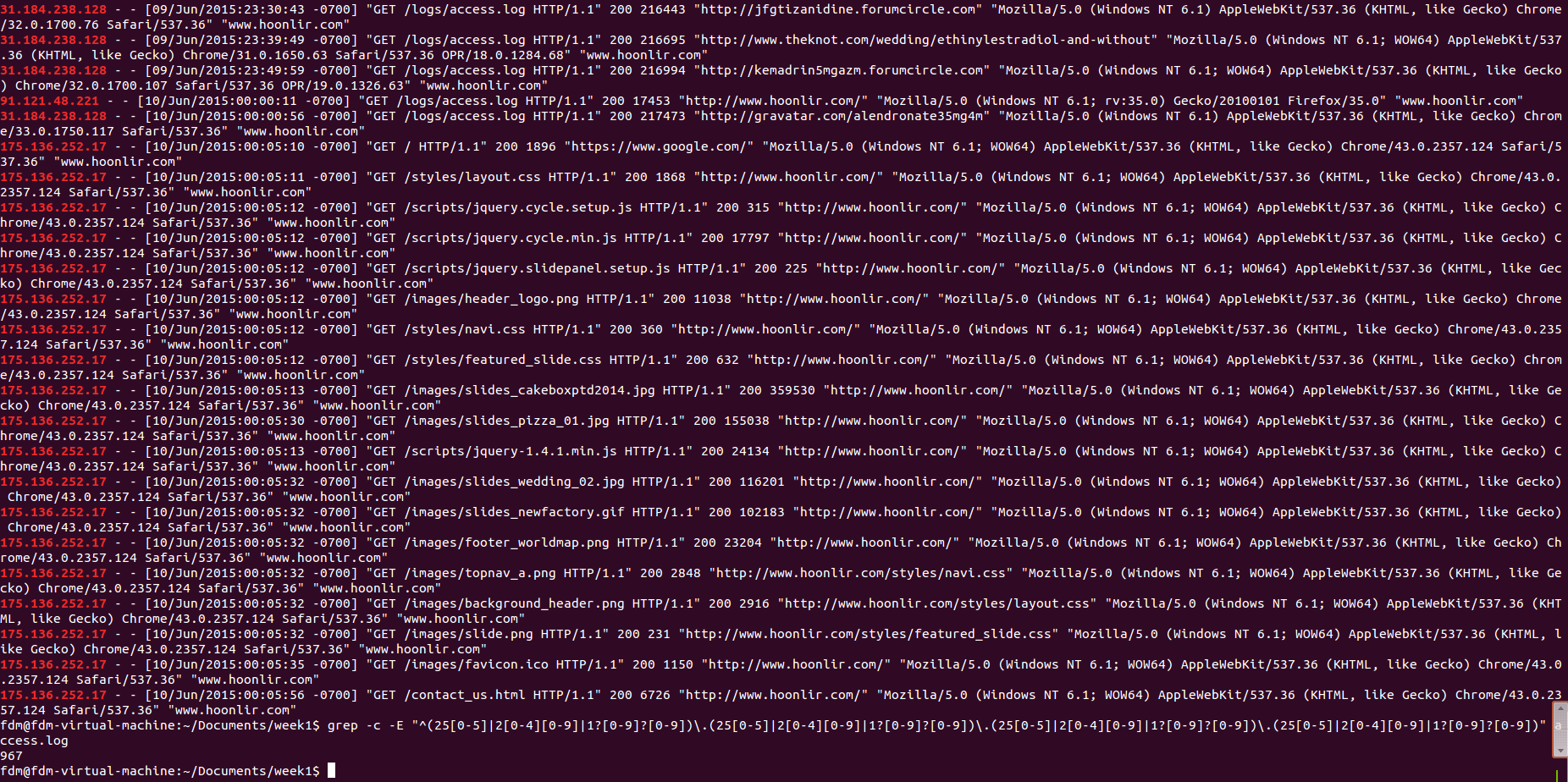
I just had to use -n to add line number and -c for count

Credit Task 1.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 1.2.1

#### Solution



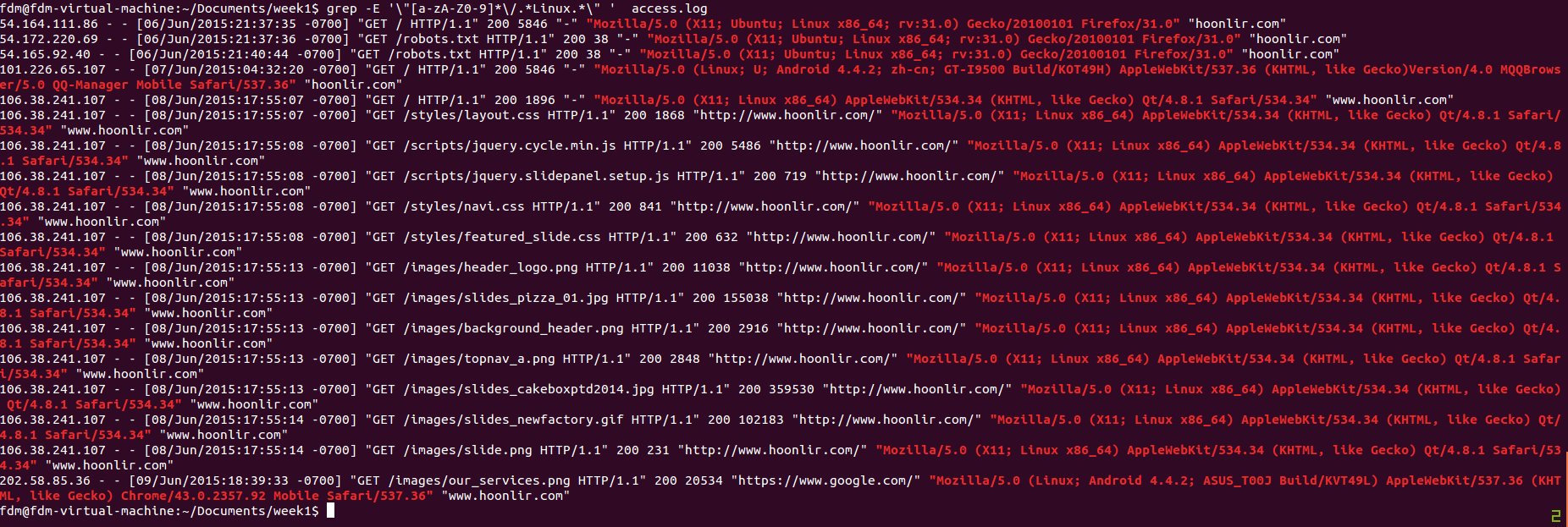
#### Comments/findings

grep -E “^(25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])\.(25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])\.(25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])\.(25[0-5]|2[0-4][0-9]|1?[0-9]?[0-9])” access.log

this was the expression I used to find the 967 IP Addresses.

## SUB Task 1.2.2

#### Solution





#### Comments/findings

The first one shows all the highlighted part and the second image shows the count.

Pass Task 2.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **10214608** |

## SUB Task 2.1.1

#### Solution



#### Comments/findings

This was rather straight forward to do, also referencing the example from task 2 helped.

## SUB Task 2.1.2

#### Solution

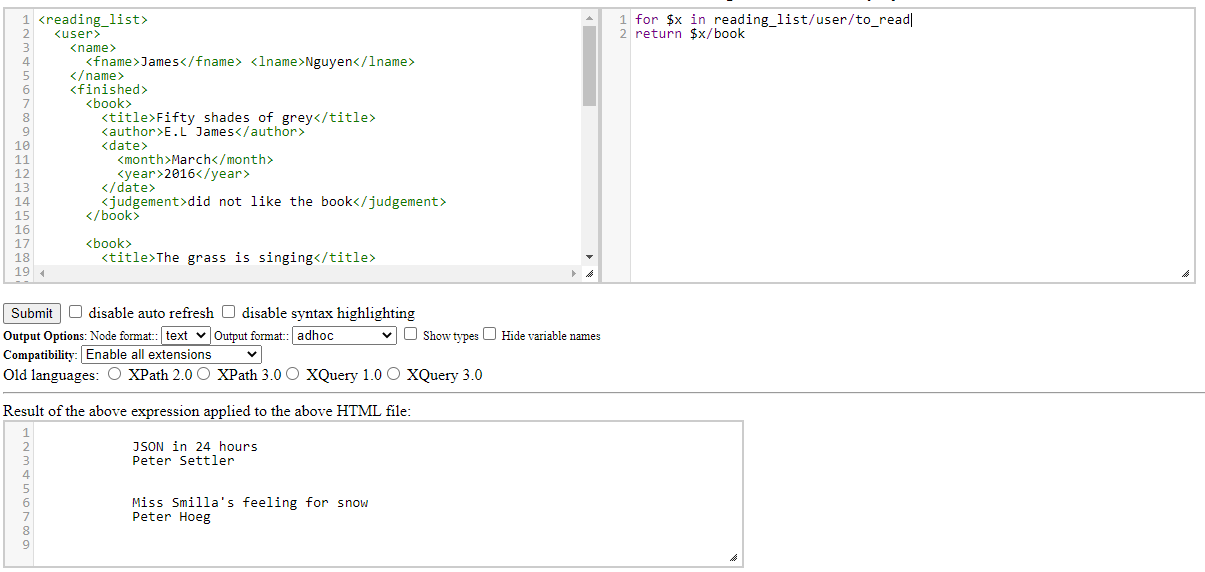


#### Comments/findings

I fixed my code after a feedback from the lecturer, my code before was “//book” now I changed it to “//book/title” which prints out all the titles for the book which was actually asked for in the subtask.

## SUB Task 2.1.3

#### Solution



#### Comments/findings

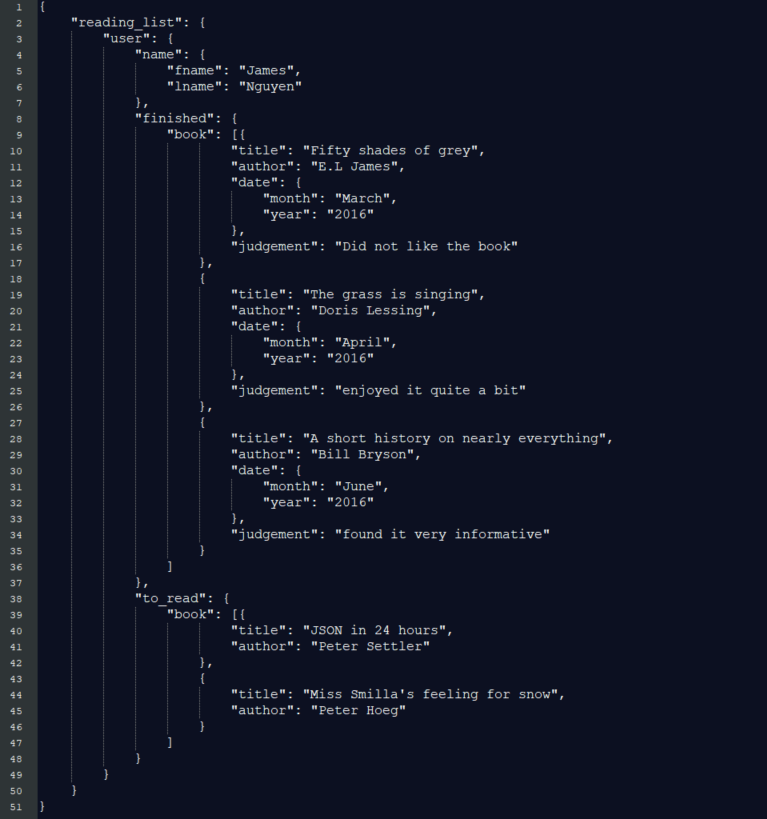
So I set my path “reading\_list/user/to\_read” to a variable “x” then just used “x” to shorten my path. I did get my desired output, but I am not sure if this is the actual way to use flwor or did I just fluke it.

Credit Task 2.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 2.2.1

#### Solution



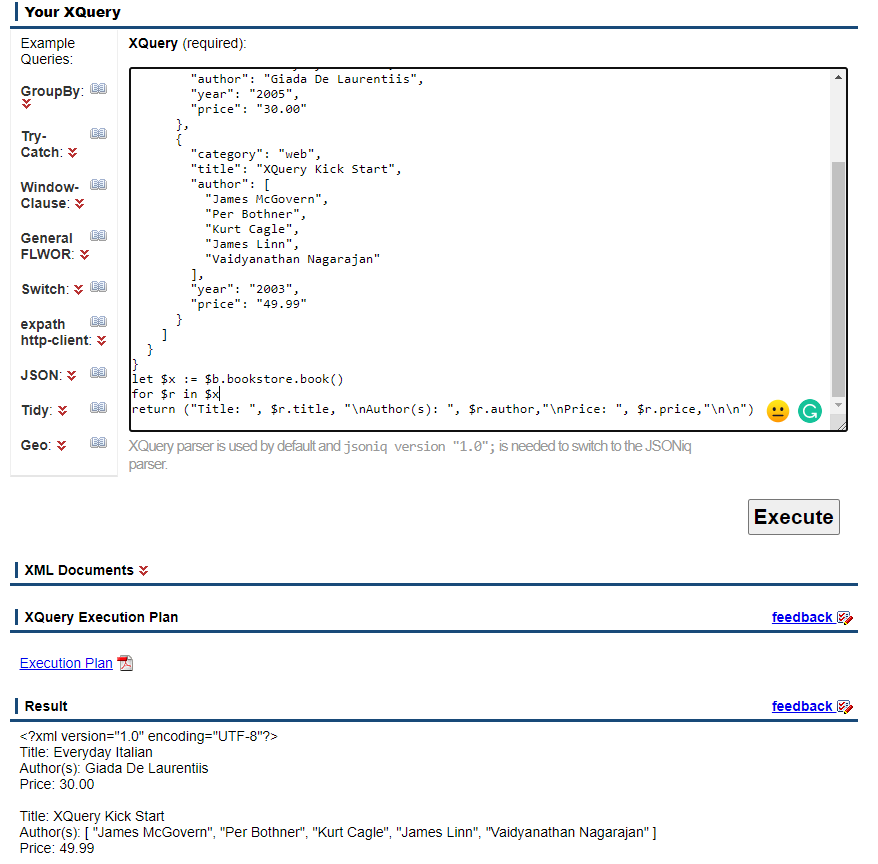


#### Comments/findings

This was rather straight forward and easy to convert to json notation took me only a couple of minutes to get used to the syntaxes

## SUB Task 2.2.2

#### Solution



#### Comments/findings

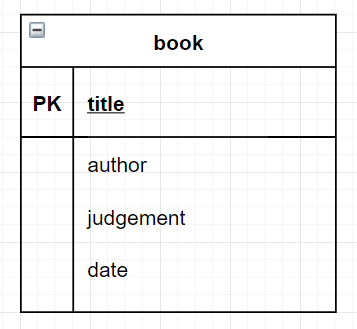
This was kind of confusing to me and took some time to figure out how to do, I had to visit a couple of websites and look through a few notes before I figured it out.

Pass Task 3.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 3.1.1

#### Solution



#### Comments/findings

(Where applicable)

## SUB Task 3.1.2

#### Solution

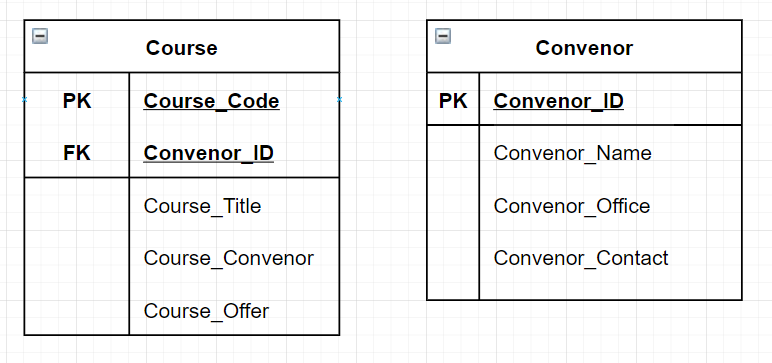
There is no one best primary key in this table, we need a composite key to uniquely identify rows in this table. So **TCode** will be the primary key while **Term** is the composite key, this will help us best identify a row in this table.

#### Comments/findings

(Where applicable)

## SUB Task 3.1.3

#### Solution



#### Comments/findings

(Where applicable)

## SUB Task 3.1.4

#### Solution

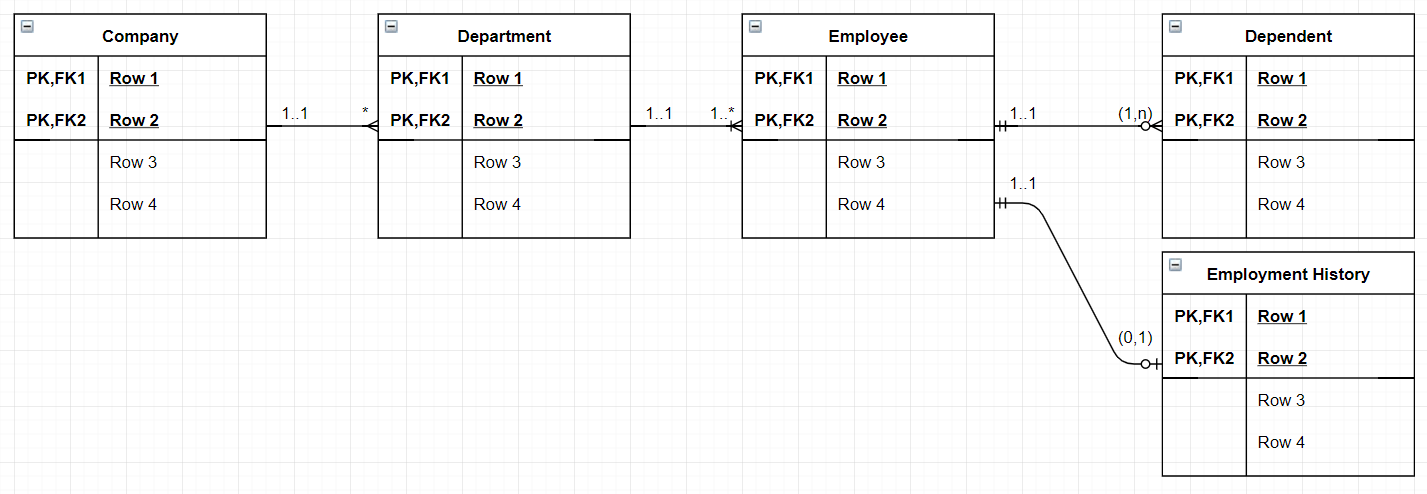
1. Text or varchar(255)
2. Client\_NO and BorrowDated

#### Comments/findings

1. Text because the text string type can hold a total of 255 characters and varchar(255) does the same thing. And 255 because most names are shorter than 255 and its more than sufficient to hold any name in it without issue.

## SUB Task 3.1.5

#### Solution

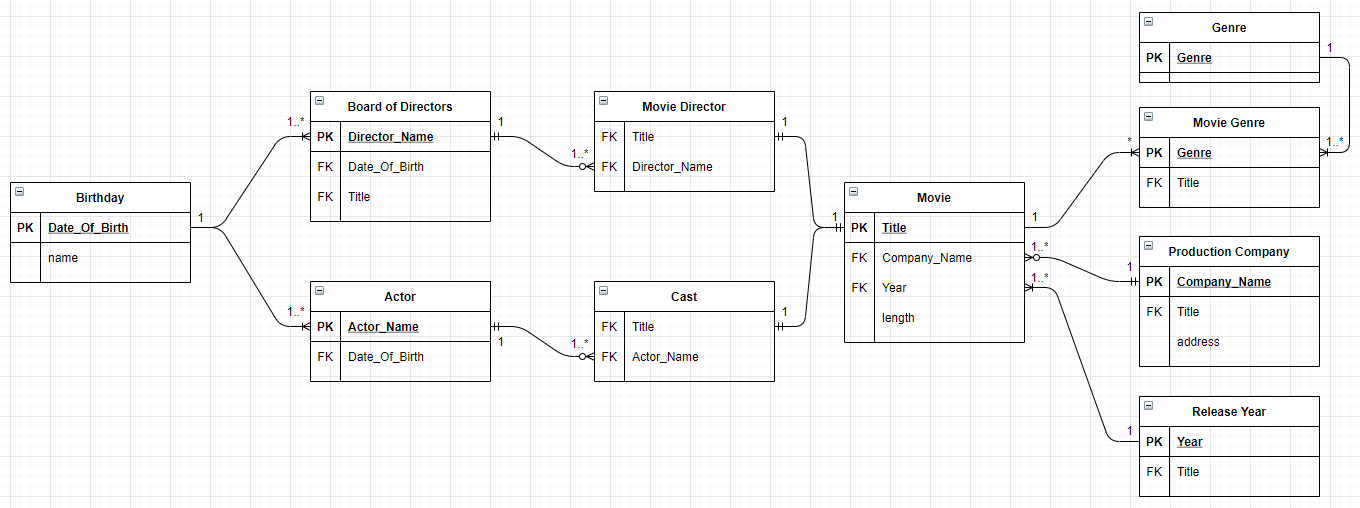


#### Comments/findings

(Where applicable)

## SUB Task 3.1.6

#### Solution



#### Comments/findings

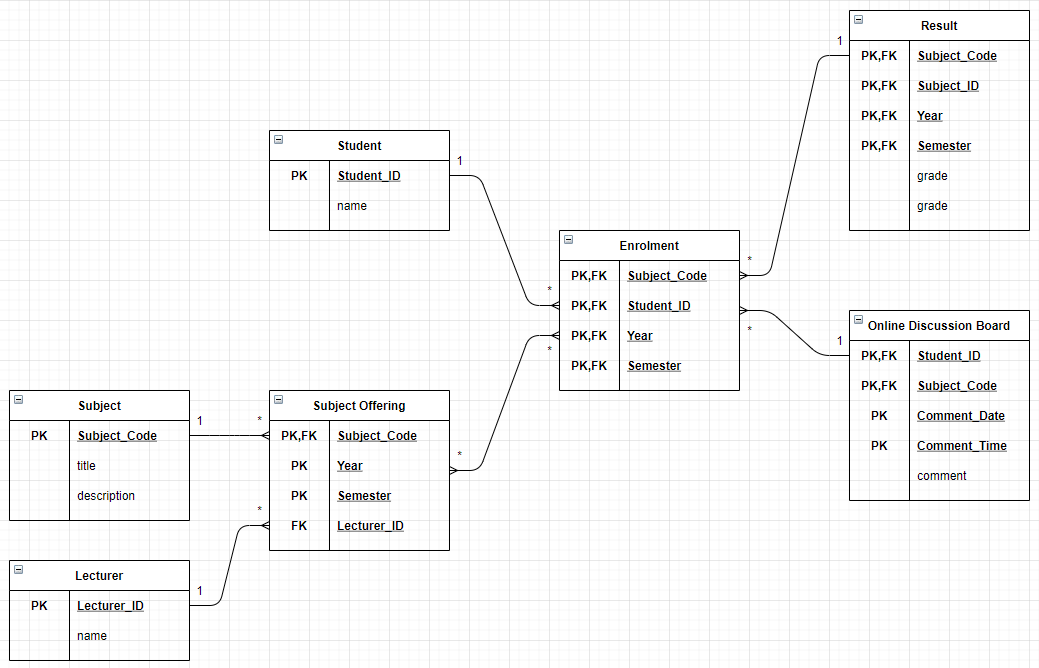
(Where applicable)

Credit Task 3.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 3.2.1

#### Solution



#### Comments/findings

(Where applicable)

Pass Task 4.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 4.1.1

#### Solution

1. The table has repeating groups.
2. Separate the values of the columns into multiple rows

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CustId | CustName | TourID | TourDesc | TourDate | StaffID | StaffName | StaffRating | Meal |
| 1040 | Sam Wilson | M304 | Bako National | 22/4/2018 | 121 | Rahim | 7 | 5 |
| 1040 | Sam Wilson | M312 | Mount Santubong | 13/5/2018 | 143 | Yap | 9 | 8 |
| 1040 | Sam Wilson | M312 | Mount Santubong | 13/5/2019 | 143 | Yap |  |  |
| 1132 | Nurul Huda Rahmat | M323 | Biking Around Kuching | 31/3/2018 | 204 | James | 8 | 4 |
| 1132 | Nurul Huda Rahmat | M312 | Mount Santubong | 13/5/2018 | 143 | Yap | 9 | 8 |

#### Comments/findings

The groups have been separated into multiple rows so it’s in the First Form.

## SUB Task 4.1.2

#### Solution

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CustId | TourID | TourDesc | TourDate | StaffID | StaffName | StaffRating | Meal |
| 1040 | M304 | Bako National | 22/4/2018 | 121 | Rahim | 7 | 5 |
| 1040 | M312 | Mount Santubong | 13/5/2018 | 143 | Yap | 9 | 8 |
| 1040 | M312 | Mount Santubong | 13/5/2019 | 143 | Yap |  |  |
| 1132 | M323 | Biking Around Kuching | 31/3/2018 | 204 | James | 8 | 4 |
| 1132 | M312 | Mount Santubong | 13/5/2018 | 143 | Yap | 9 | 8 |

|  |  |
| --- | --- |
| CustId | CustName |
| 1040 | Sam Wilson |
| 1040 | Sam Wilson |
| 1040 | Sam Wilson |
| 1132 | Nurul Huda Rahmat |
| 1132 | Nurul Huda Rahmat |

#### Comments/findings

For a table to be in 2nd Normal Form there must be no partial dependency, in the table above the primary keys are **CustID** and **TourDate**, and every column is either totally dependent on both or none expect **CustName**. CustName is only dependent on CustID so it is partially dependent thus it has been removed and moved to a separate table.

## SUB Task 4.1.3

#### Solution

|  |  |
| --- | --- |
| CustID | CustName |
| 1040 | Sam Wilson |
| 1132 | Nurul Huda Rahmat |

|  |  |
| --- | --- |
| TourID | TourDesc |
| M304 | Bako National |
| M312 | Mount Santubong |
| M323 | Biking Around Kuching |

|  |  |  |  |
| --- | --- | --- | --- |
| StaffID | StaffName | StaffRating | Meal |
| 121 | Rahim | 7 | 5 |
| 143 | Yap | 9 | 8 |
| 204 | James | 8 | 4 |

#### Comments/findings

For a table to be in the 3rd Normal Form there must be no transitive dependency, from the table in the 2NF we can see that there are a few columns that does not depend on the primary key(s) those columns have been separated into multiple other tables, which can be seen above.

Credit Task 4.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 4.2.1

#### Solution

|  |  |  |
| --- | --- | --- |
| **Dependency** | **Possible (Yes/No)** | **Why/why not?** |
| **A -> B** | **No** | A has repeating values |
| **A -> C** | **No** | A has repeating values |
| **A -> D** | **No** | A has repeating values |
| **B -> A** | **No** | B has repeating values |
| **B -> C** | **No** | B has repeating values |
| **B -> D** | **No** | B has repeating values |
| **C -> A** | **Yes** | Value of C is unique |
| **C -> B** | **Yes** | Value of C is unique |
| **C -> D** | **Yes** | Value of C is unique |
| **{A, B} -> C** | **Yes** | Value of A, B combined is unique |
| **{A, B} -> D** | **Yes** | Value of A, B combined is unique |
| **{B, C} -> A** | **Yes** | Value of B, C combined is unique |
| **{B, C} -> D** | **Yes** | Value of B, C combined is unique |
| **{C, D} -> A** | **Yes** | Value of C, D combined is unique |
| **{C, D} -> B** | **Yes** | Value of C, D combined is unique |
| **{A, C} -> B** | **Yes** | Value of A, C combined is unique |
| **{A, C} -> D** | **Yes** | Value of A, C combined is unique |

#### Comments/findings

No, we do need a composite key because a single column is sufficient to uniquely identify a relation. For this case C or D can be used as primary keys as they are unique and can uniquely identity the relation

Pass Task 5.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 5.1.1

#### Solution

Room

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| RoomNo | Int(3) |
| PricePerNight | Double(10,2) |
| RoomType | Varchar(30) |

CustomerFeedback

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| CommentDate | Varchar(10) |
| CustomerID | Int(10) |
| RoomNo | Int(3) |
| DateCheckIn | Varchar(10) |
| Comments | Varchar(500) |

RoomBooking

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| CustomerID | Int(10) |
| RoomNo | Int(3) |
| DateCheckIn | Varchar(10) |
| DateCheckOut | Varchar(10) |
| CustSpecialRequest | Varchar(500) |

Payment

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| CustomerID | Int(10) |
| RoomNo | Int(3) |
| DateCheckIn | Varchar(10) |
| Amount | Double(10,2) |
| PaymentDate | Varchar(10) |
| PaymentMethod | Varchar(30) |

Customer

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| CustomerID | Int(10) |
| CustName | Varchar(30) |
| CustPhoneNo | Int(10) |
| CustEmail | Varchar(30) |
| CustDOB | Varchar(10) |

#### Comments/findings

## SUB Task 5.1.2

#### Solution

|  |  |
| --- | --- |
| **Relation** | **Primary Key** |
| Room | RoomNo |
| CustomerFeedback | CustomerID, RoomNo, CommentDate |
| RoomBooking | CustomerID, RoomNo, DateCheckIn |
| Payment | CustomerID, RoomNo, DateCheckIn |
| Customer | CustomerID |

#### Comments/findings

This Table shows what is the primary key for the corresponding table, if there are more than one primary key its means it’s a composite key.

## SUB Task 5.1.3

#### Solution

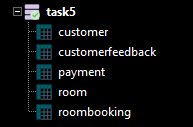
|  |  |  |
| --- | --- | --- |
| **Relation** | **Foreign Key** | **References** |
| Room |  |  |
| CustomerFeedback | CustomerID,  RoomNo,  DateCheckIn | Customer.CustomerID, Room.RoomNo, RoomBooking.DateCheckIn |
| RoomBooking | CustomerID,  RoomNo, | Customer.CustomerID,  Room.RoomNo |
| Payment | CustomerID,  RoomNo,  DateCheckIn | Customer.CustomerID,  Room.RoomNo,  RoomBooking.DateCheckIn |
| Customer |  |  |

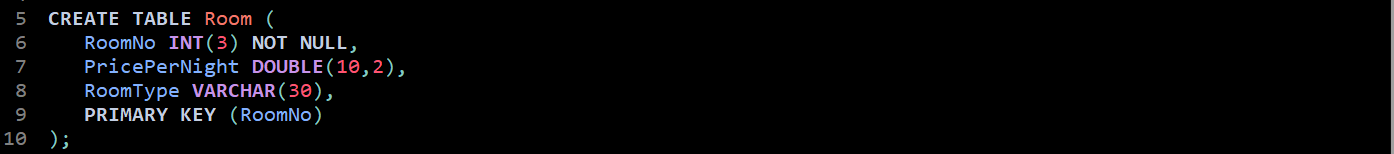
#### Comments/findings

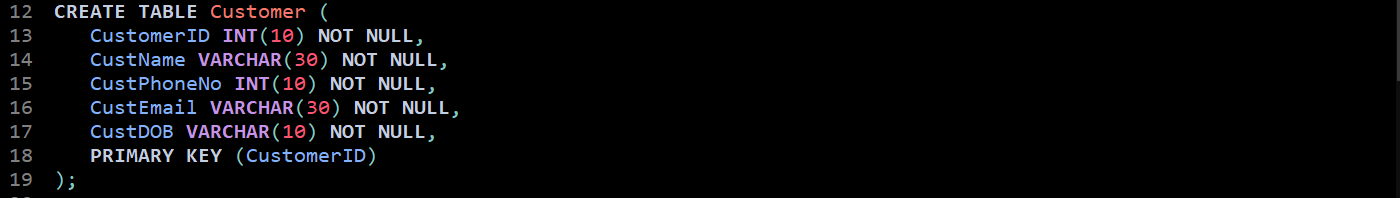
Here the table show what foreign key references what table, for example from the CustomerFeedback table CustomerID column references CustomerID column in the Customer table.

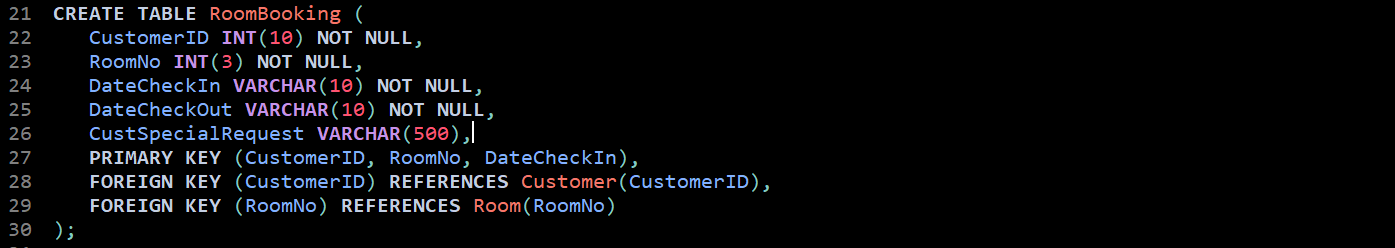
## SUB Task 5.1.4

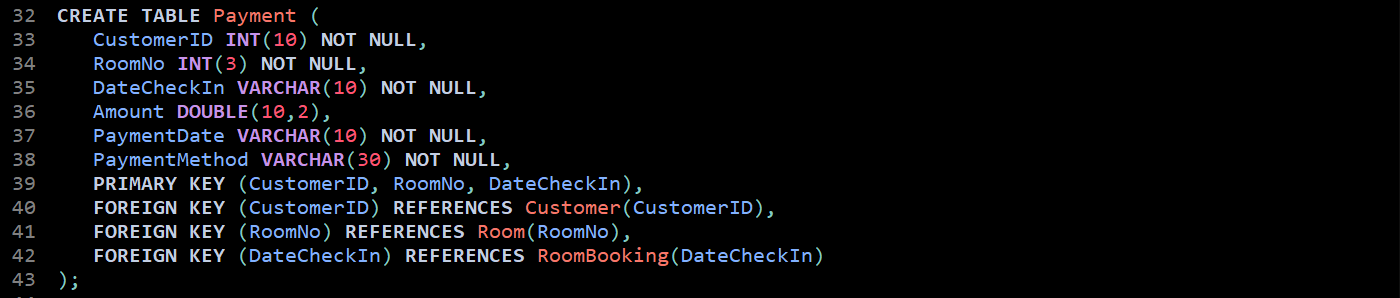
#### Solution

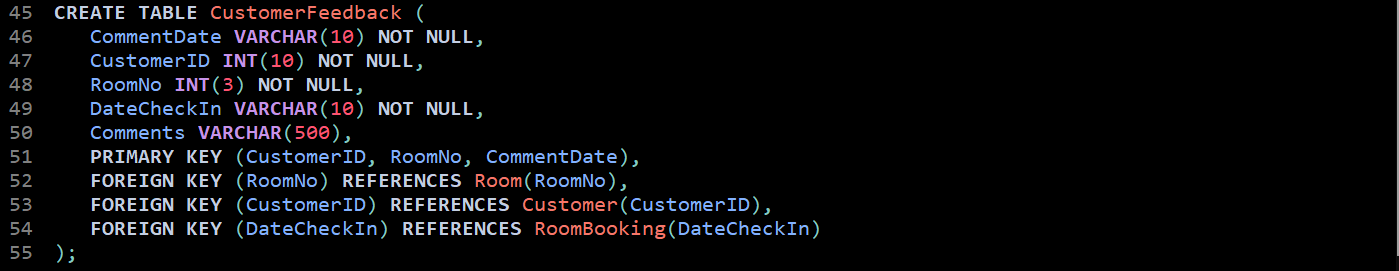












#### Comments/findings

All the codes used to create all the tables.

## SUB Task 5.1.5

#### Solution

a)



b)



#### Comments/findings

Used ALTER to modify the table, used ADD or DROP to add or remove columns from the table.

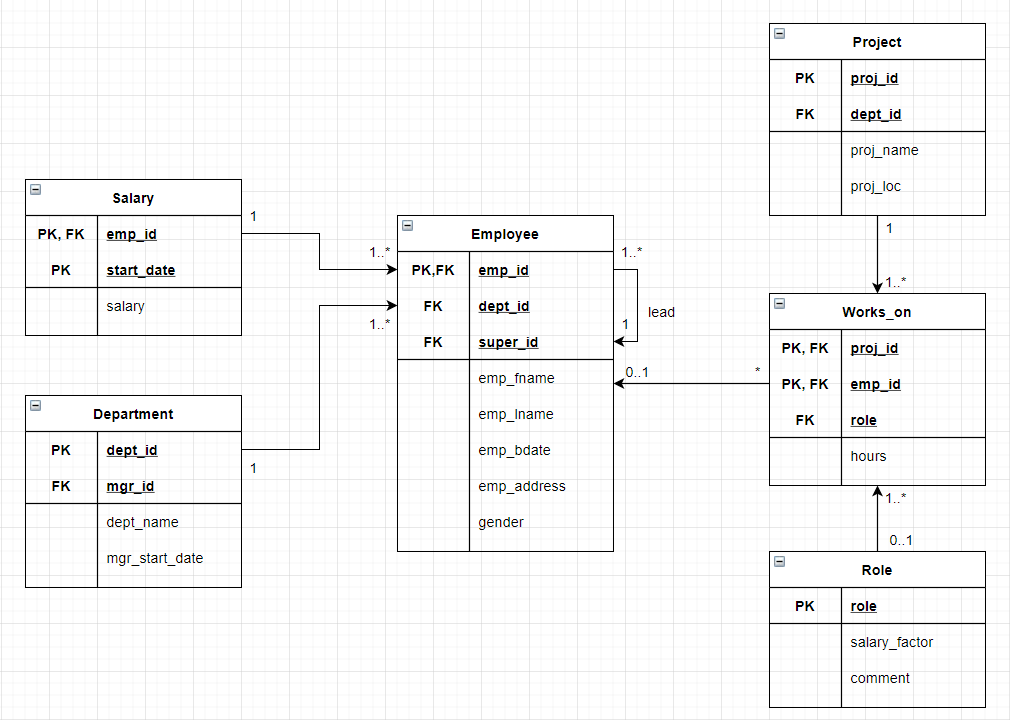
Credit Task 5.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

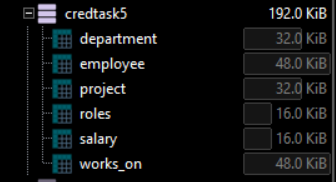
## SUB Task 5.2.1

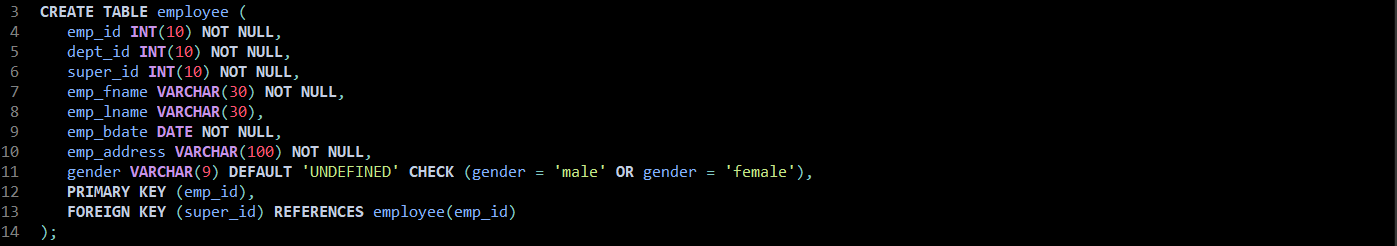
#### Solution

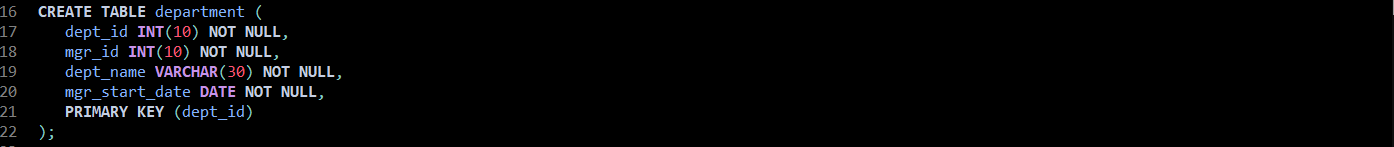
a)

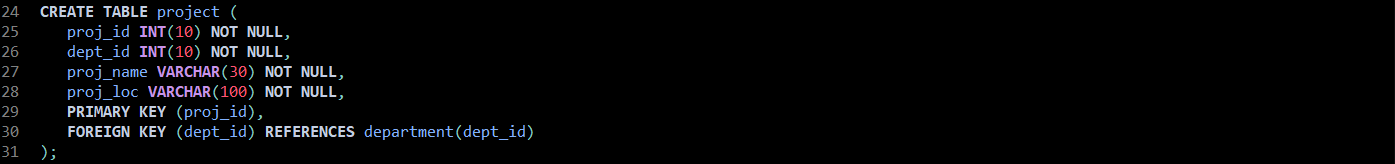


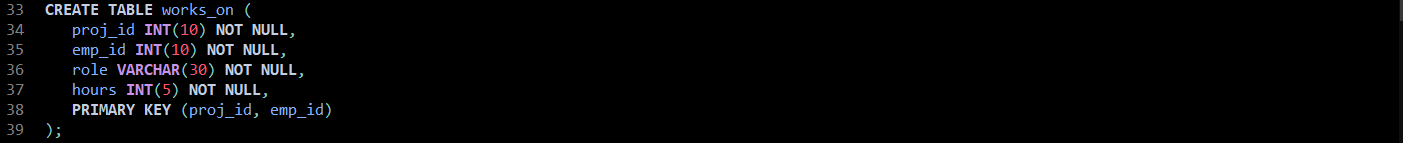
b)

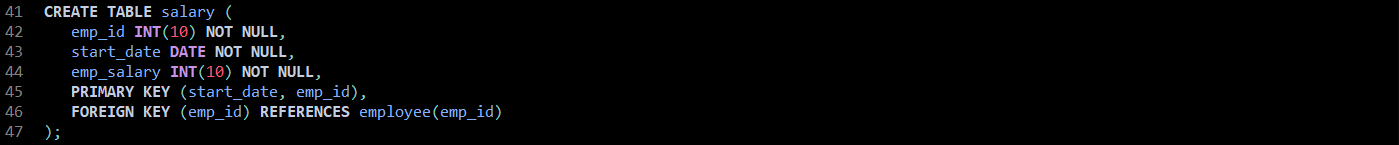


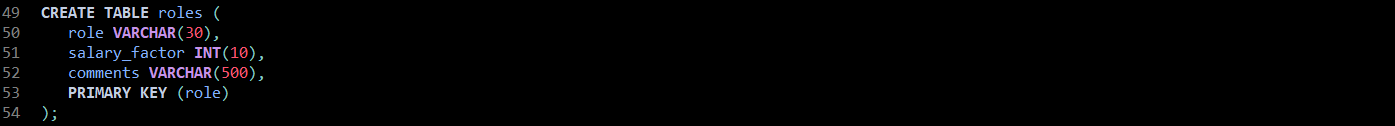


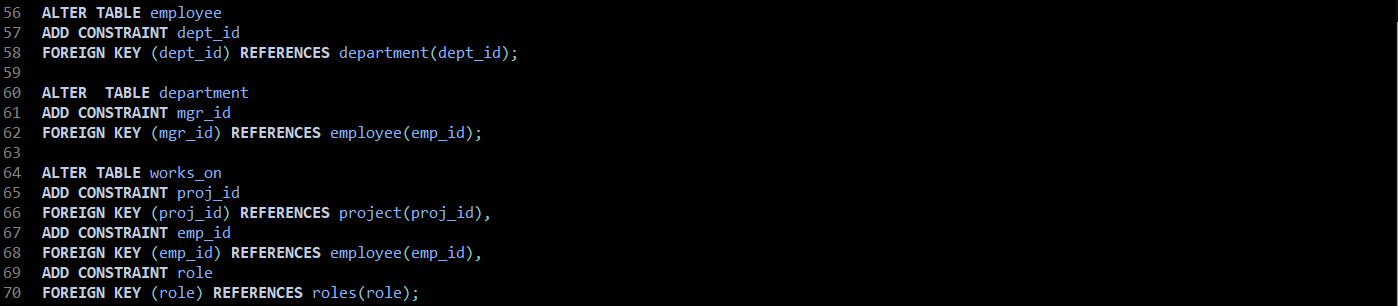


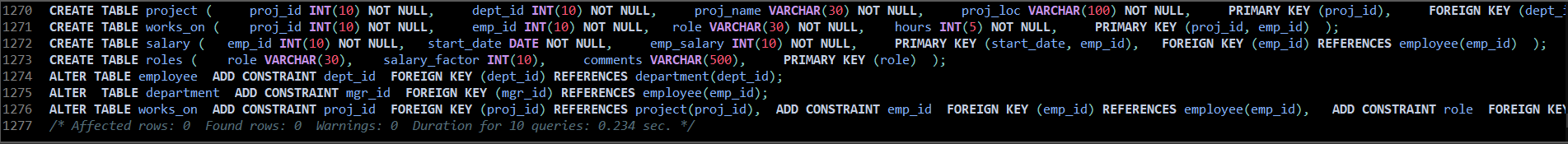












* Gender has a default value which is Undefined as some people may not want to share their gender.
* Check is used for gender so only male and female are allowed to enter if not then it will be undefined.

#### Comments/findings

…

(Where applicable)

Pass Task 6.1 report

Logo

Description automatically generated

|  |  |
| --- | --- |
| Student Name | Student ID |
| **Masrur Rahman Zahin** | **101214608** |
| **Mohammad Mobin** | **101228706** |
|  |  |

## Company background

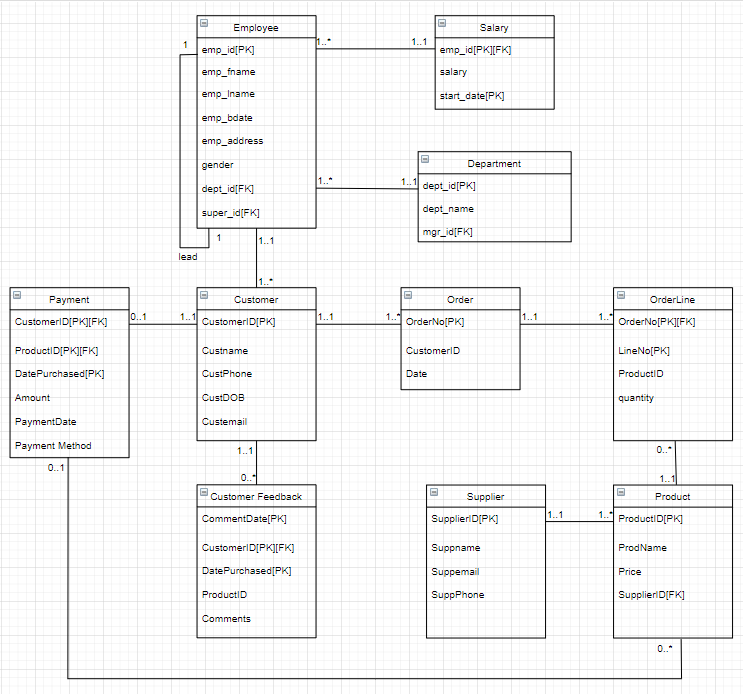
The company we chose is **H&M**, it is a Swedish multinational clothing company famous for its trendy clothing for men, women, teenagers, and children. H&M was founded in 1947 by **Erling Persson**. The first shop was opening in Vasteras, Sweden. H&M was originally called Hennes which is Swedish for “hers” and sold women’s clothing exclusively. Erling later partnered with **Mauritz Widforss**, which led to the addition of menswear collection into their clothing business and the company name was changed to **Hennes & Mauritz**, more commonly known as H&M.

Website: https://www2.hm.com/en\_us/index.html

## scenario

As with most companies, H&M has a strict employee management system as can be seen in the employee relation, each of them has specific roles regarding their occupation and belongs to a certain branch or department, some employees can be the superior of other employees hence the recursion occurs within the employee table. There is a customer entity which takes in important details the customers who are being identified by their id, they have an option to present comments or reviews based on their experience. This however is optional. Since one customer can have multiple comments, a separate entity was created called customer feedback which takes in a more detailed approach specifically for the comments, in order to achieve the third normal form. The customer has order which contains list of different products. However, this relation between order and product would be many to many, and to provide good design implementation, we introduced a weak entity which goes by the name of OrderLine, in order to eliminate the many to many relations. In this, each product has one supplier, and one supplier can supply multiple products, the suppliers are identified by their id as shown in the relation along with other contact details of the suppliers. Lastly the customer performs payment with the appropriate details taking the product and customer into consideration

## erd

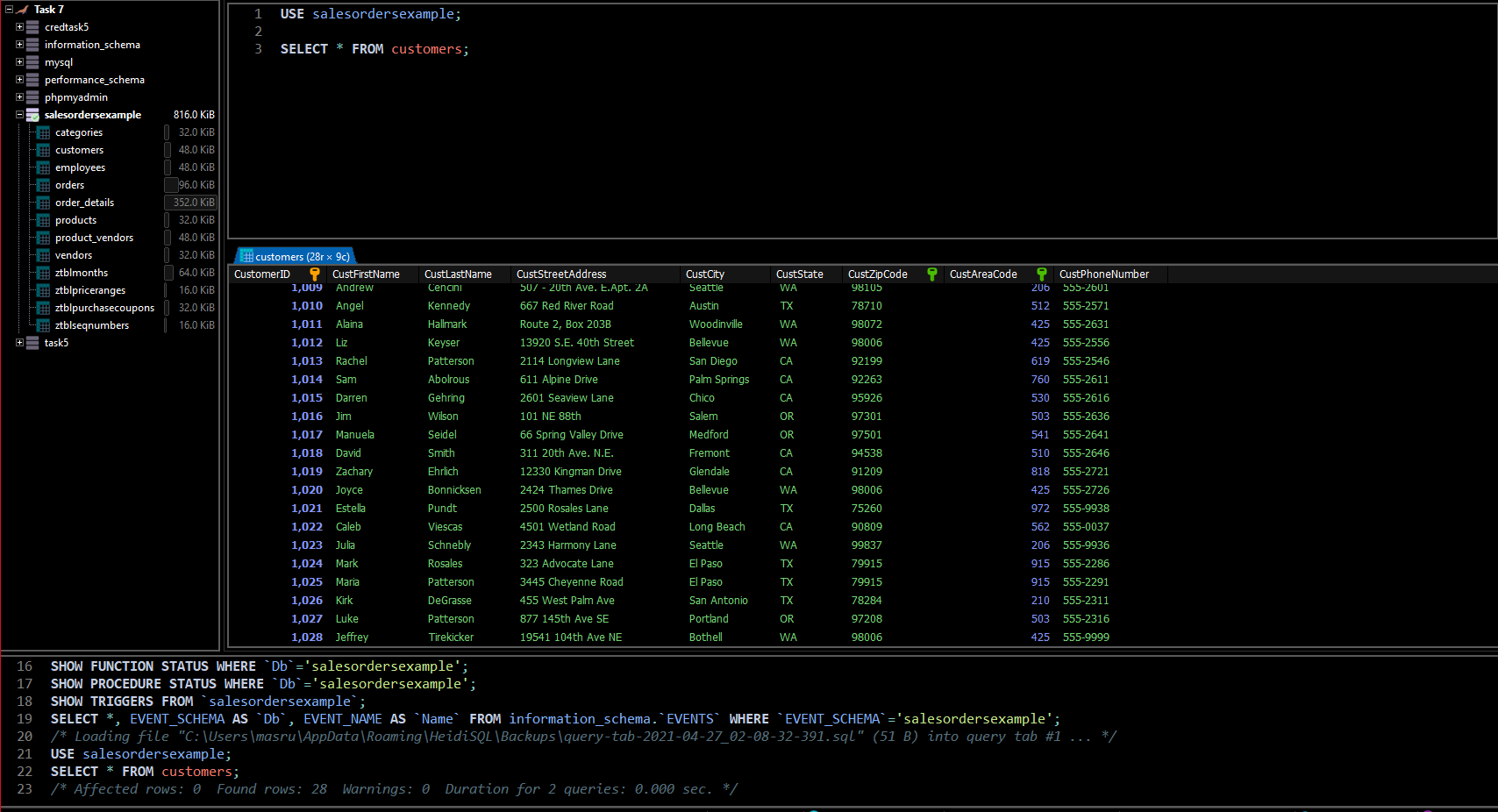
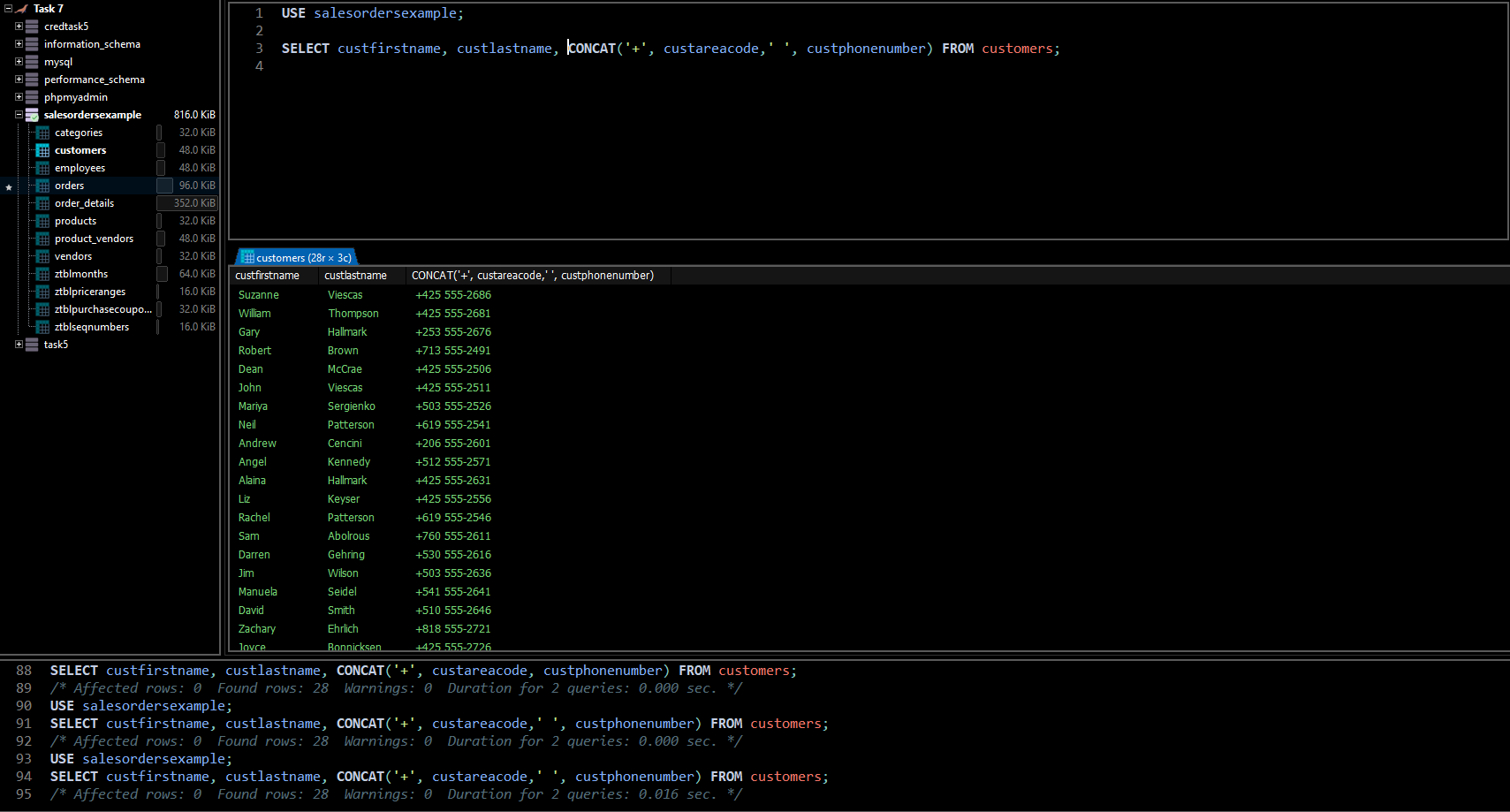


Pass Task 7.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

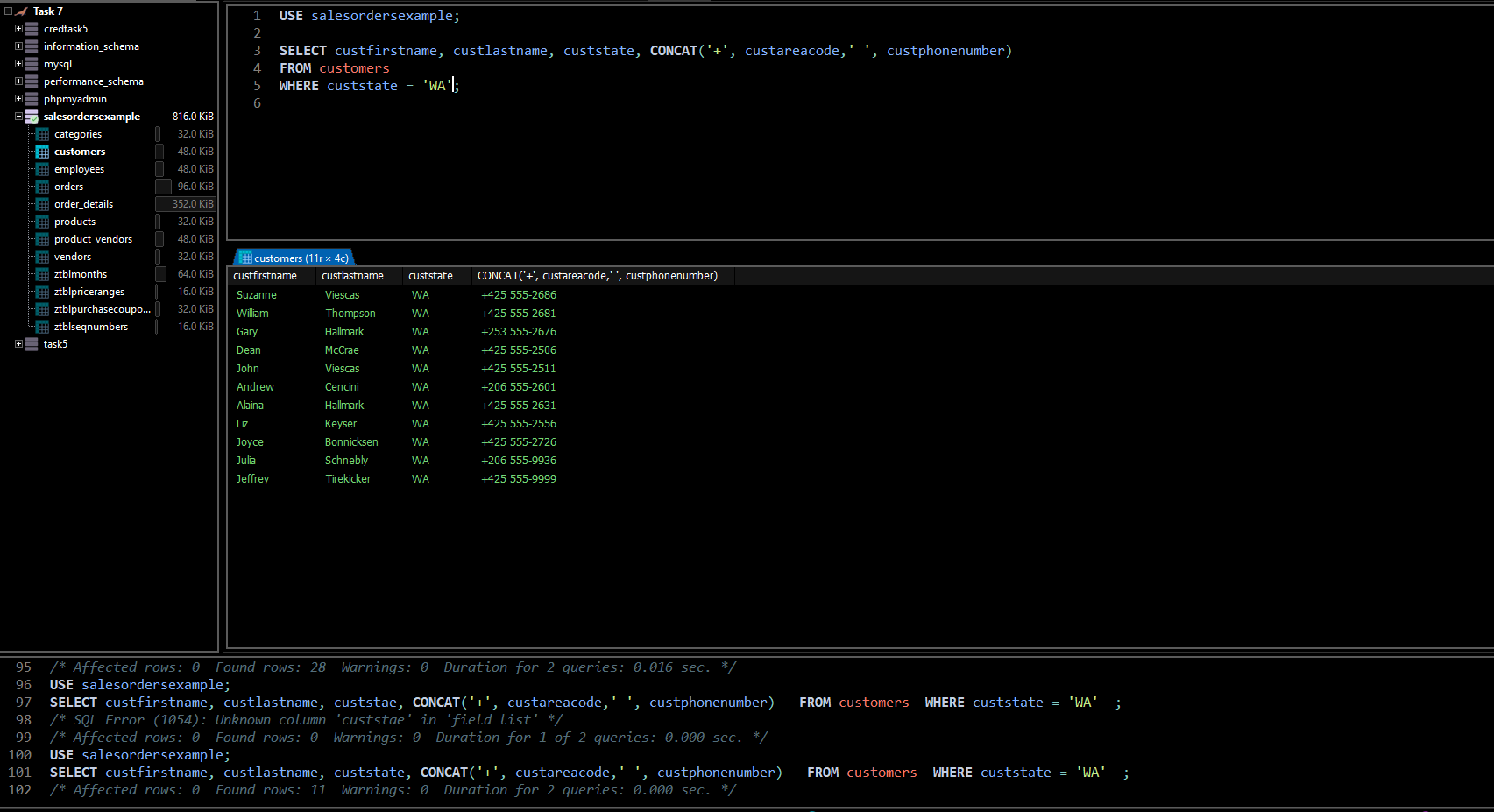
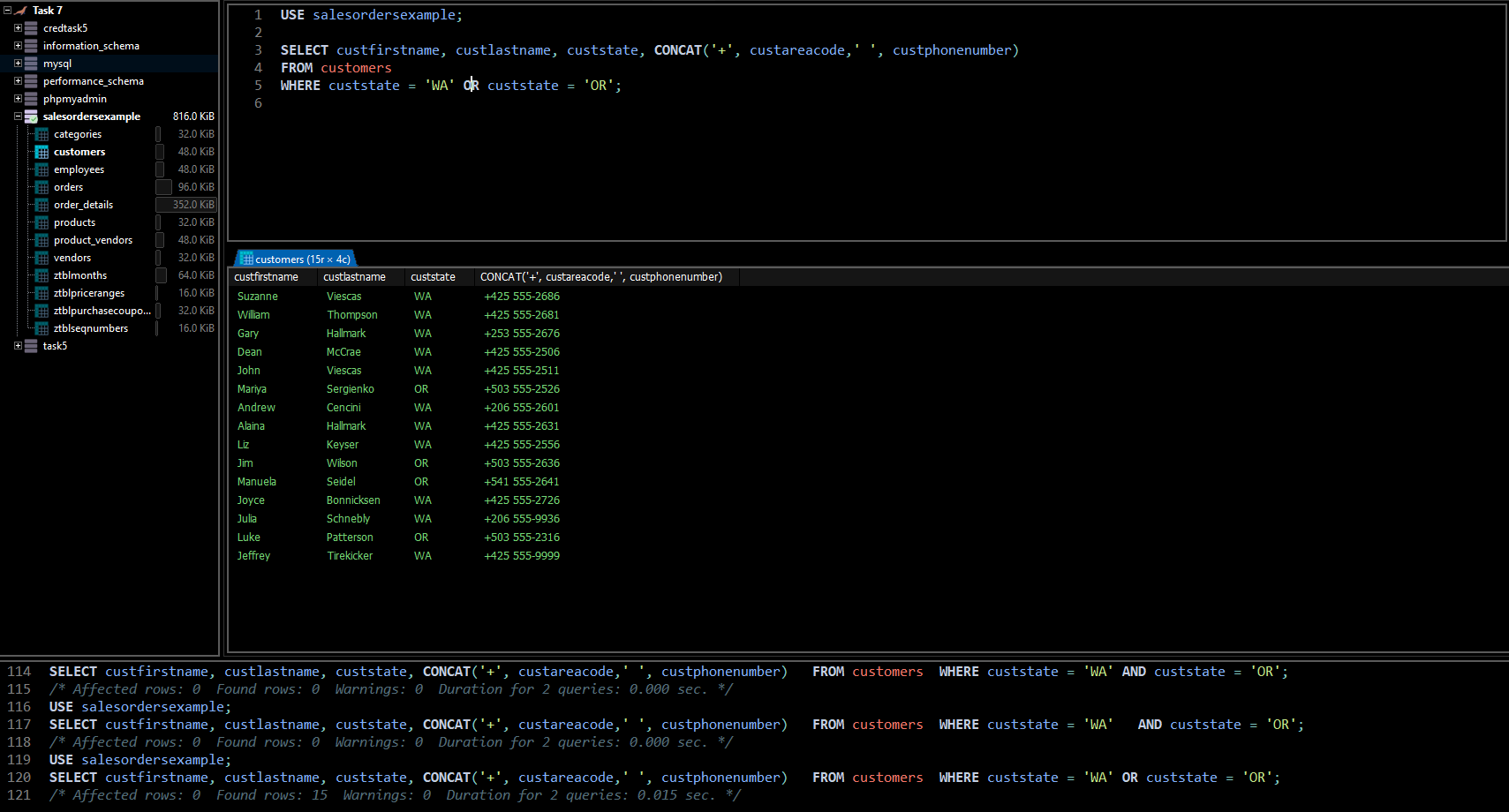
## SUB Task 7.1.1

#### Solution

1. 
2. 

## SUB Task 7.1.2

#### Solution

1. 
2. 

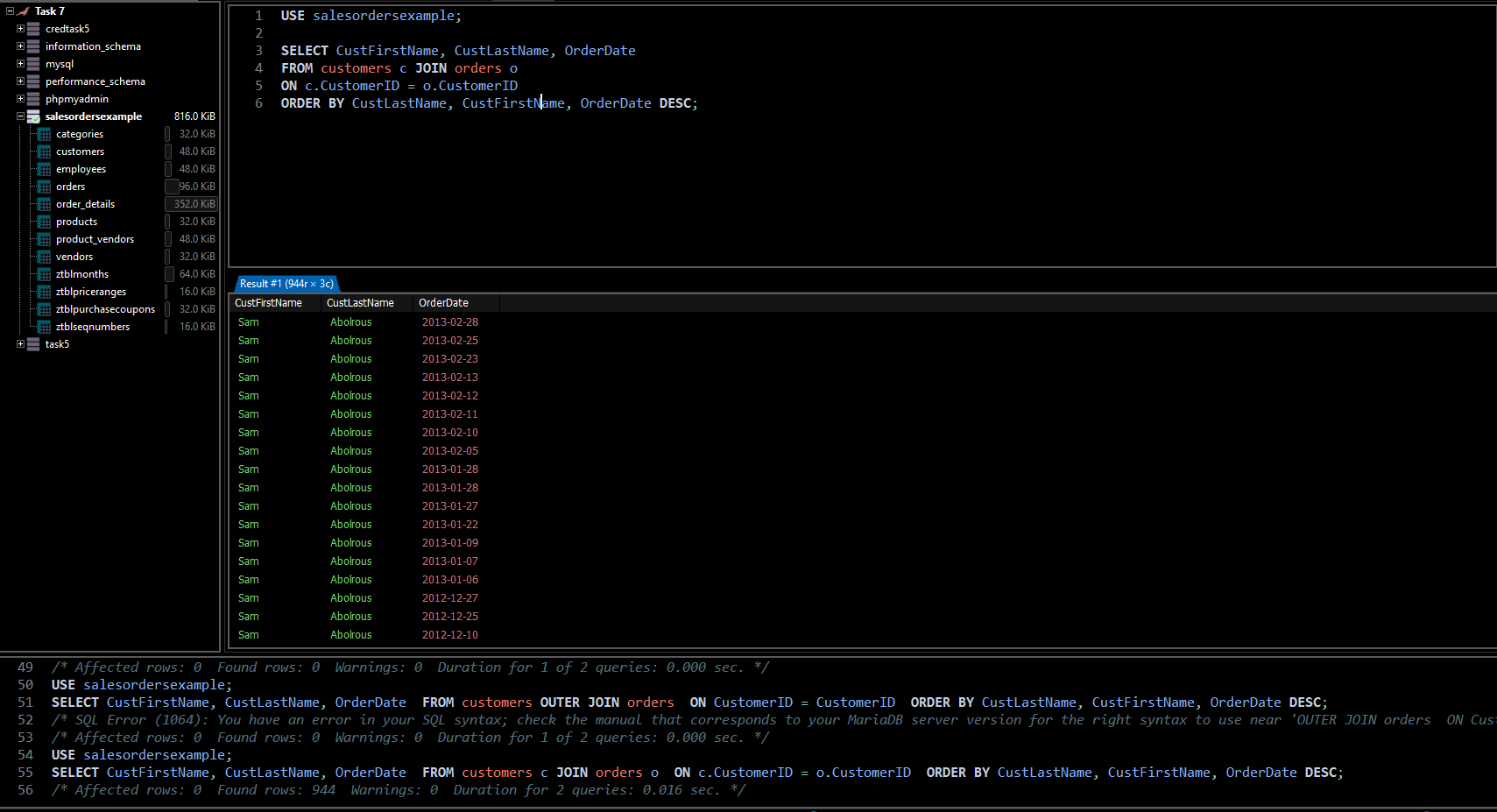
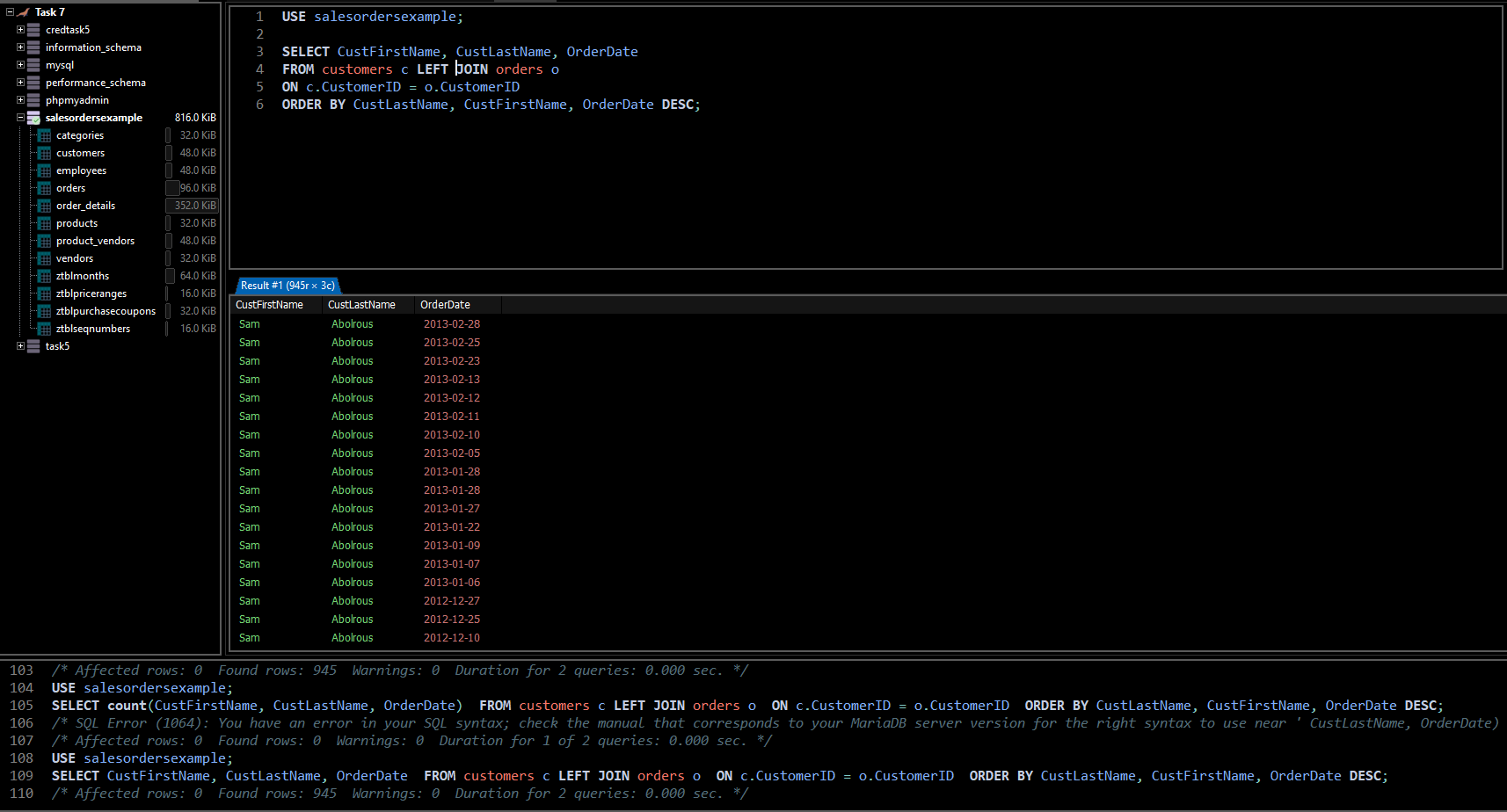
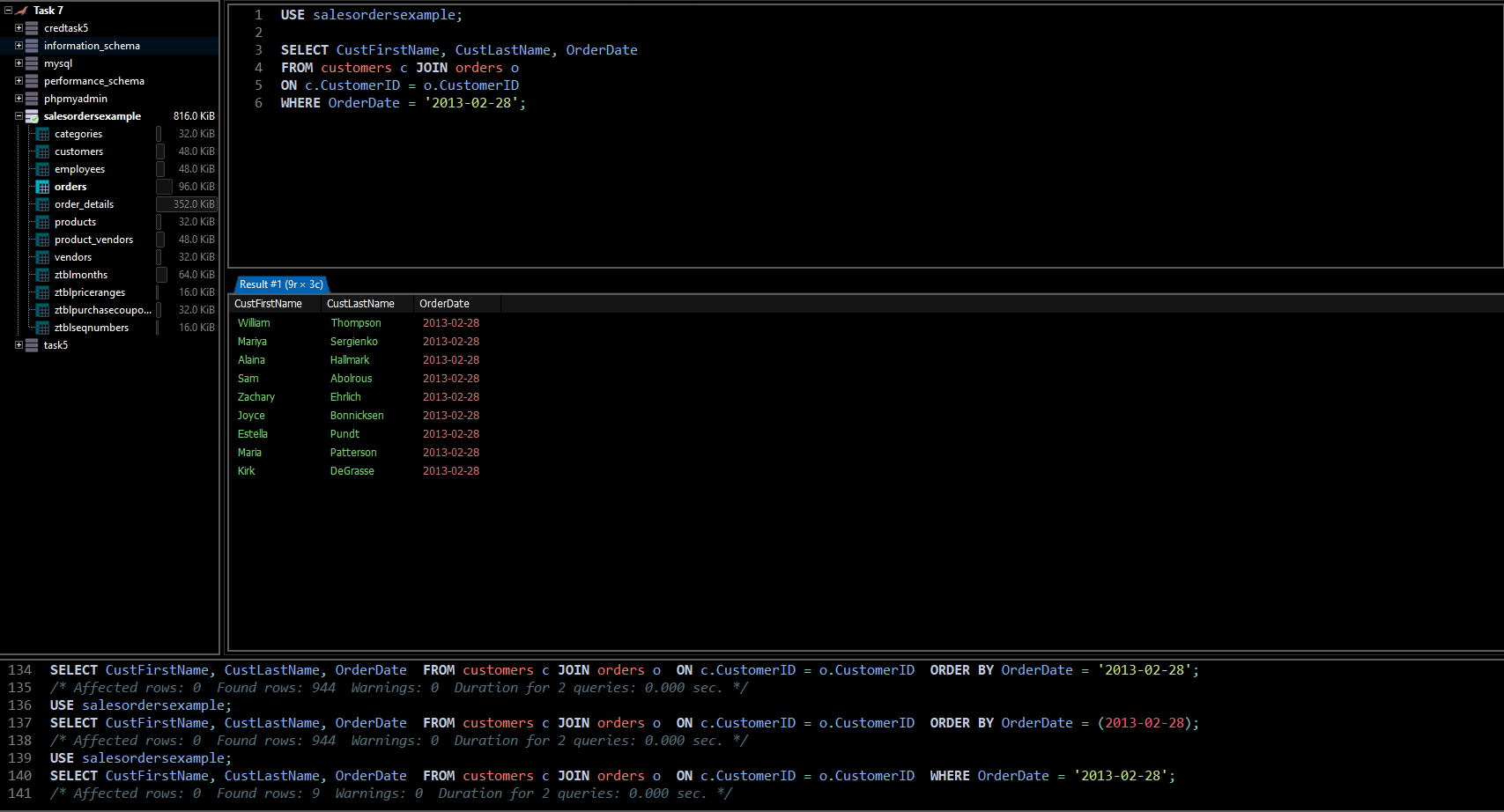
## SUB Task 7.1.3

#### Solution

#### 

## SUB Task 7.1.4

#### Solution

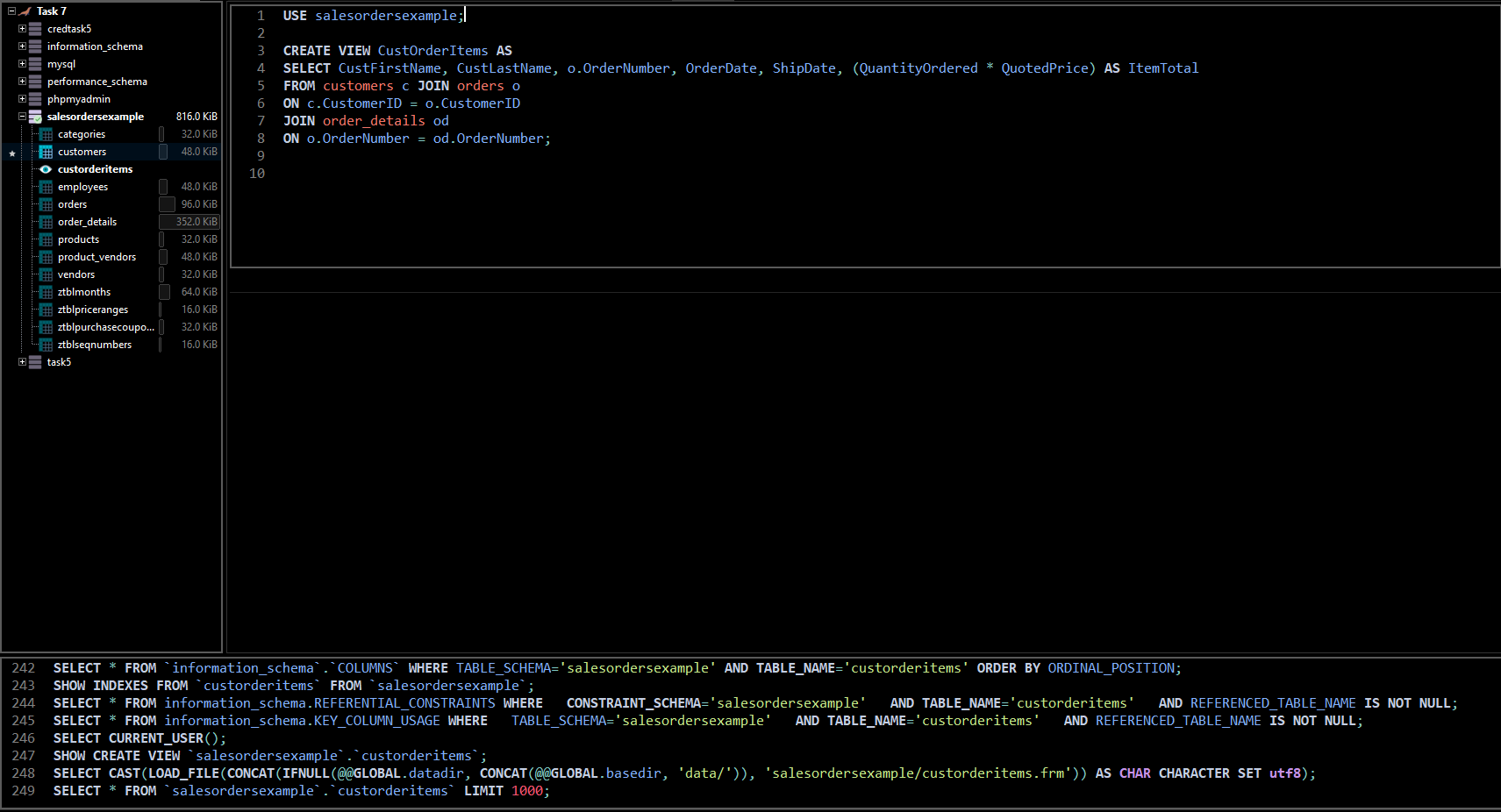
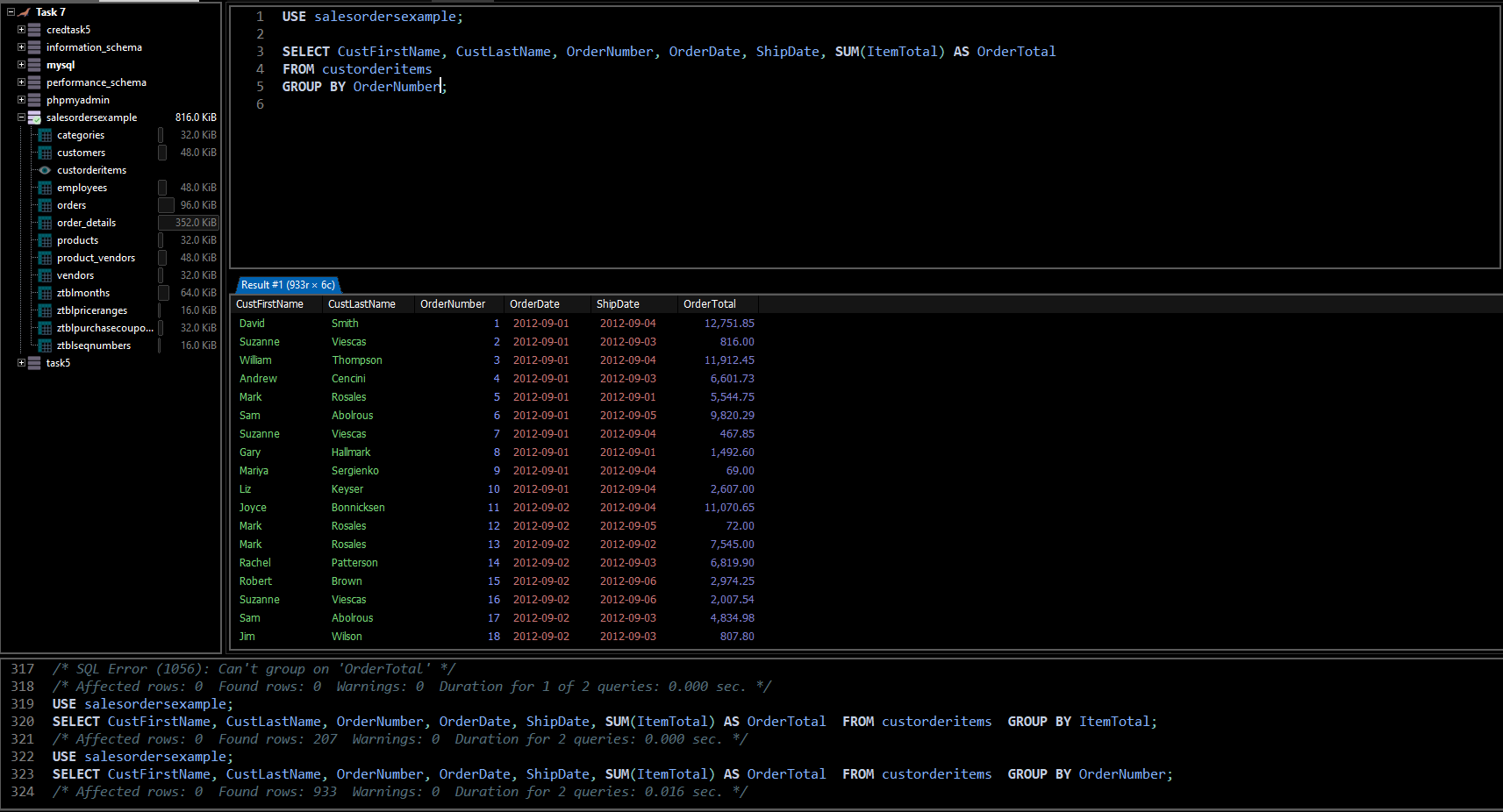
1. To check all the order dates of every product ordered by a customer.
2. 
3. 
4. 

#### Comments/findings

LEFT JOIN was used to show the one customer who was left in the customer table that didn’t show up in the orders tables because the customer had no order

## SUB Task 7.1.5

#### Solution

1. 
2. 

#### Comments/findings

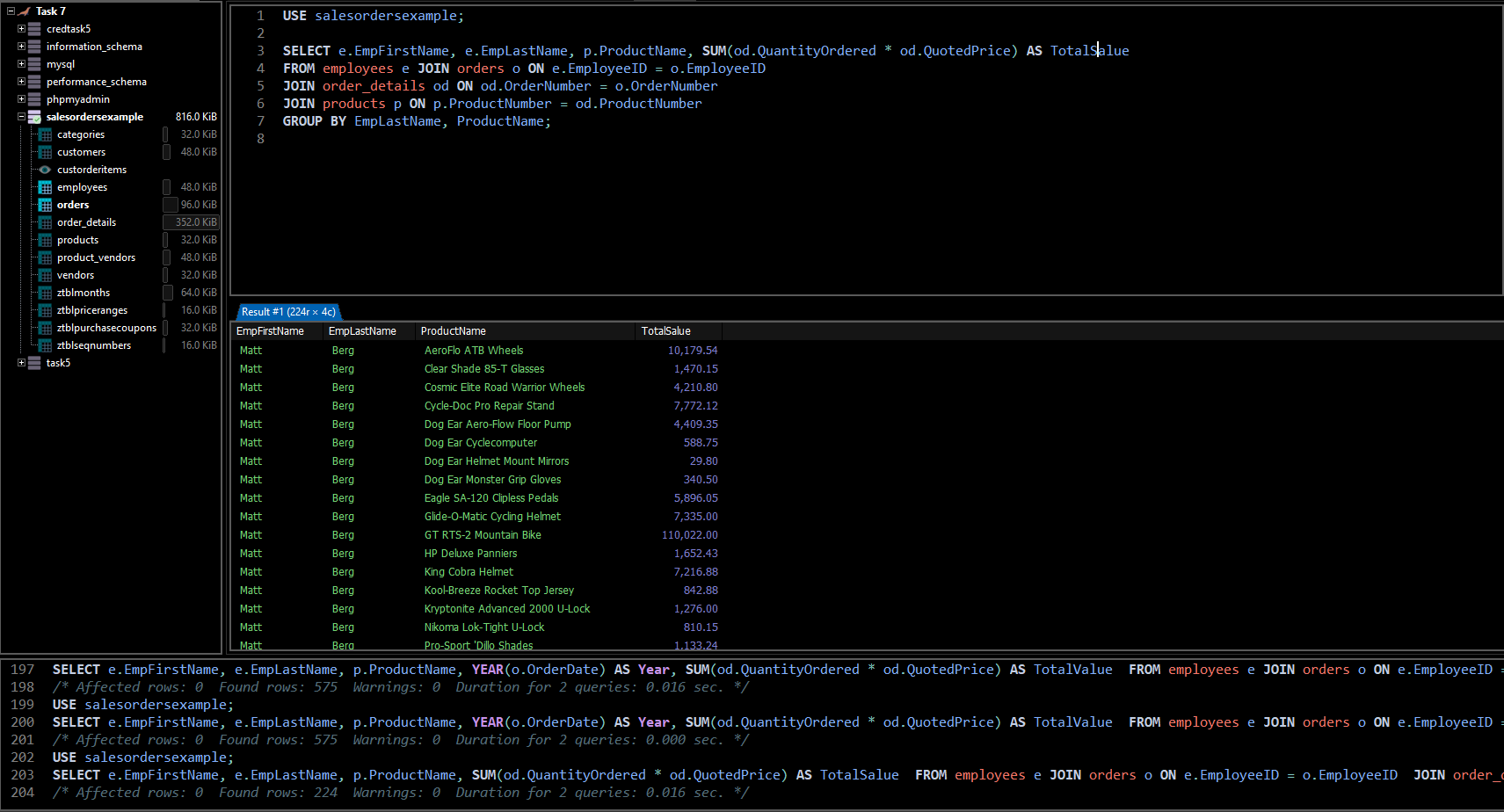
The query was written first before creating a view to minimize error.

Credit Task 7.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

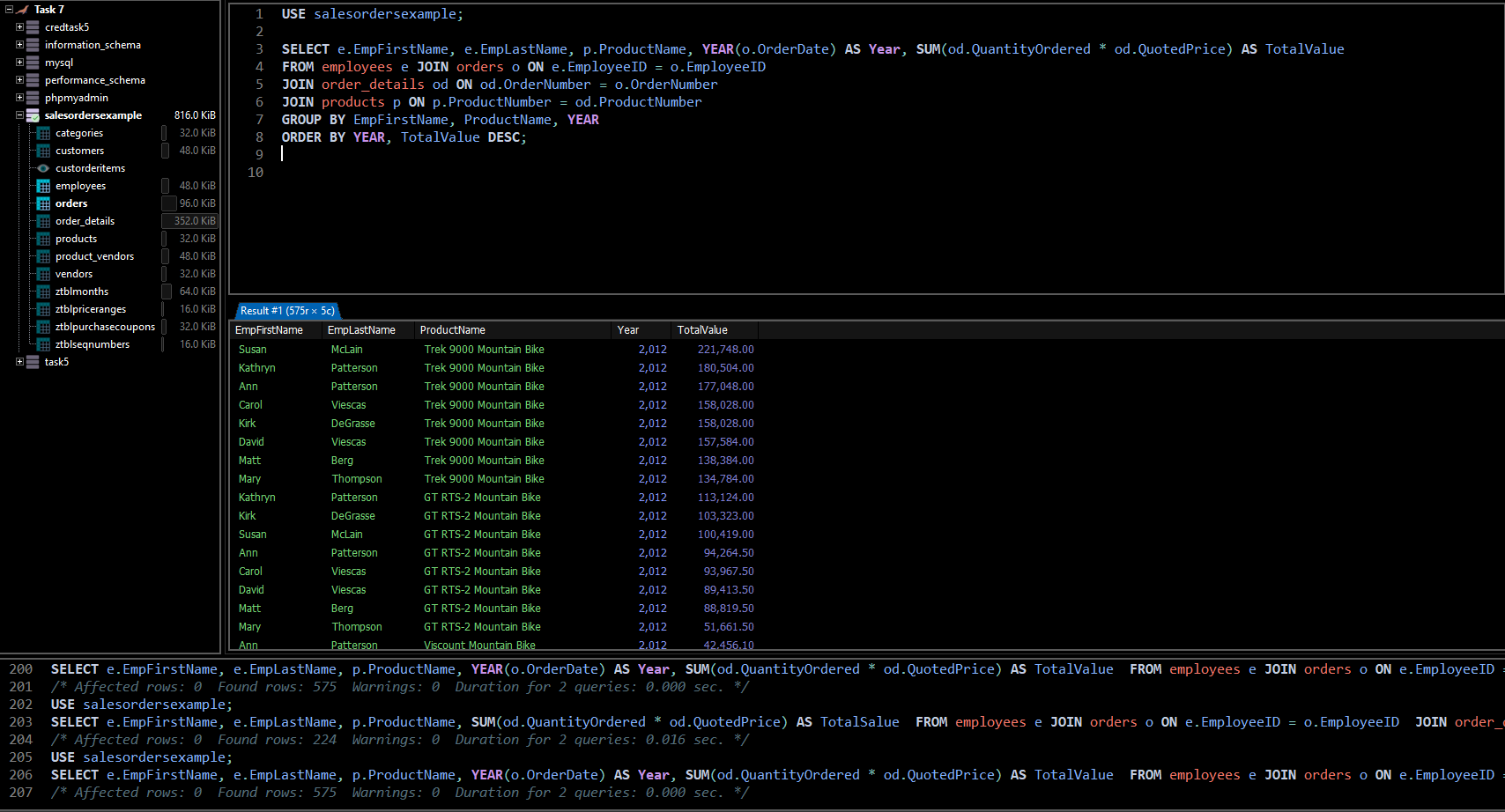
## SUB Task 7.2.1

#### Solution



## SUB Task 7.2.2

#### Solution



## SUB Task 7.2.3 - Optional

#### Solution



#### Comments/findings

…

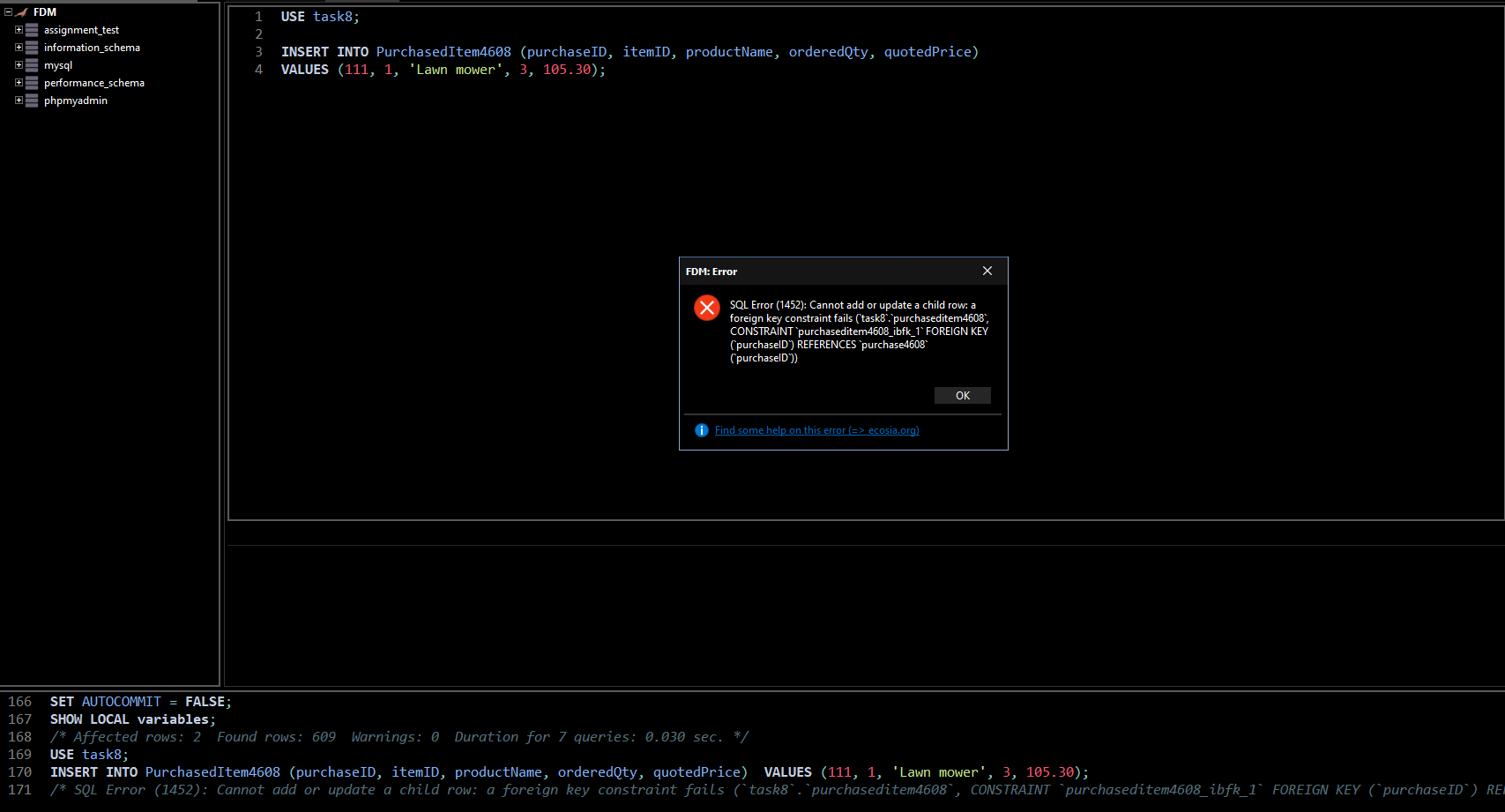
(Where applicable)

Pass Task 8.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 8.1.1

#### Solution





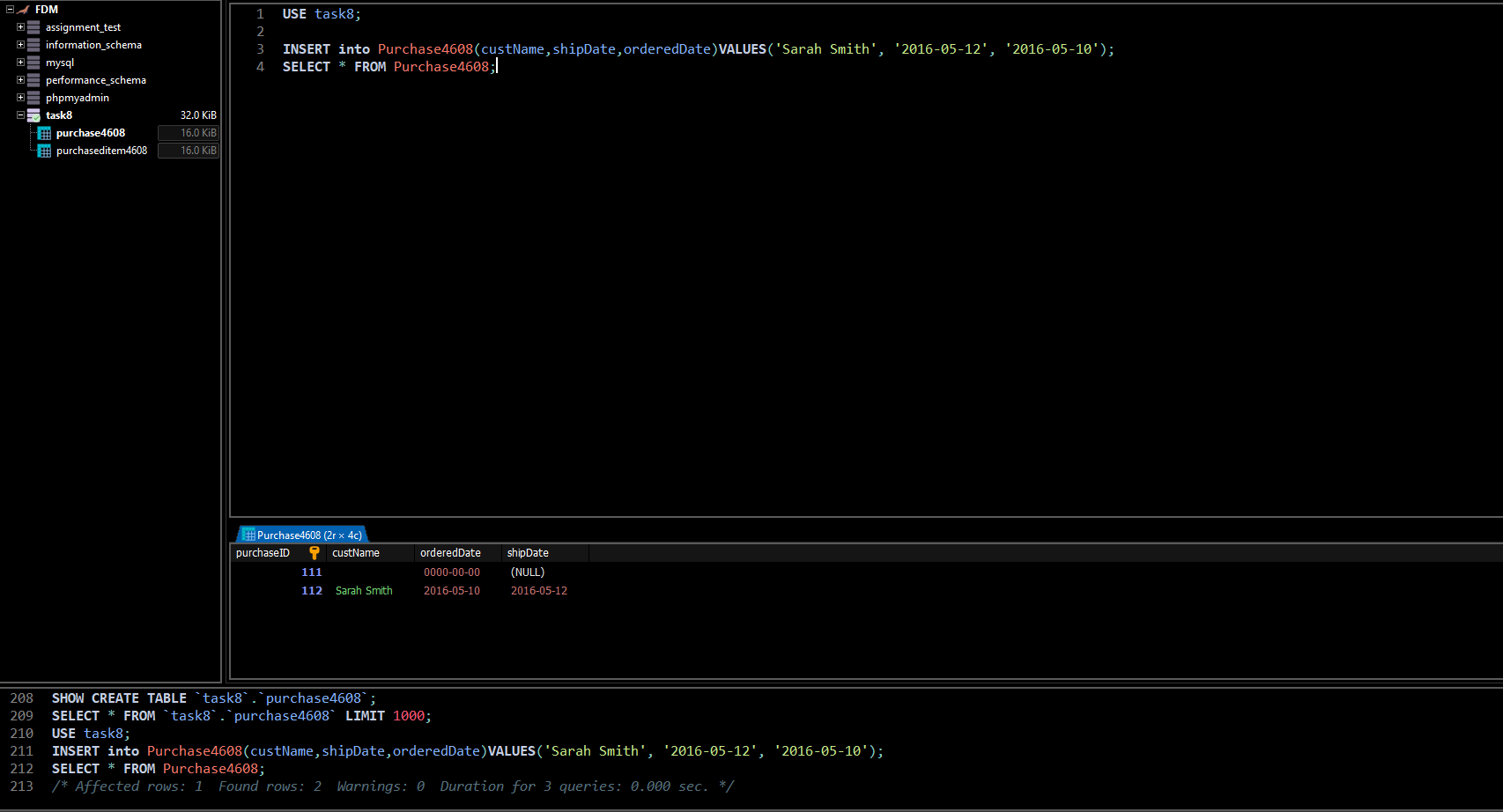
#### Comments/findings

At first we get an error because the parent table is empty, and we are trying to connect the child table to the parent table via a foreign key which does not exists.

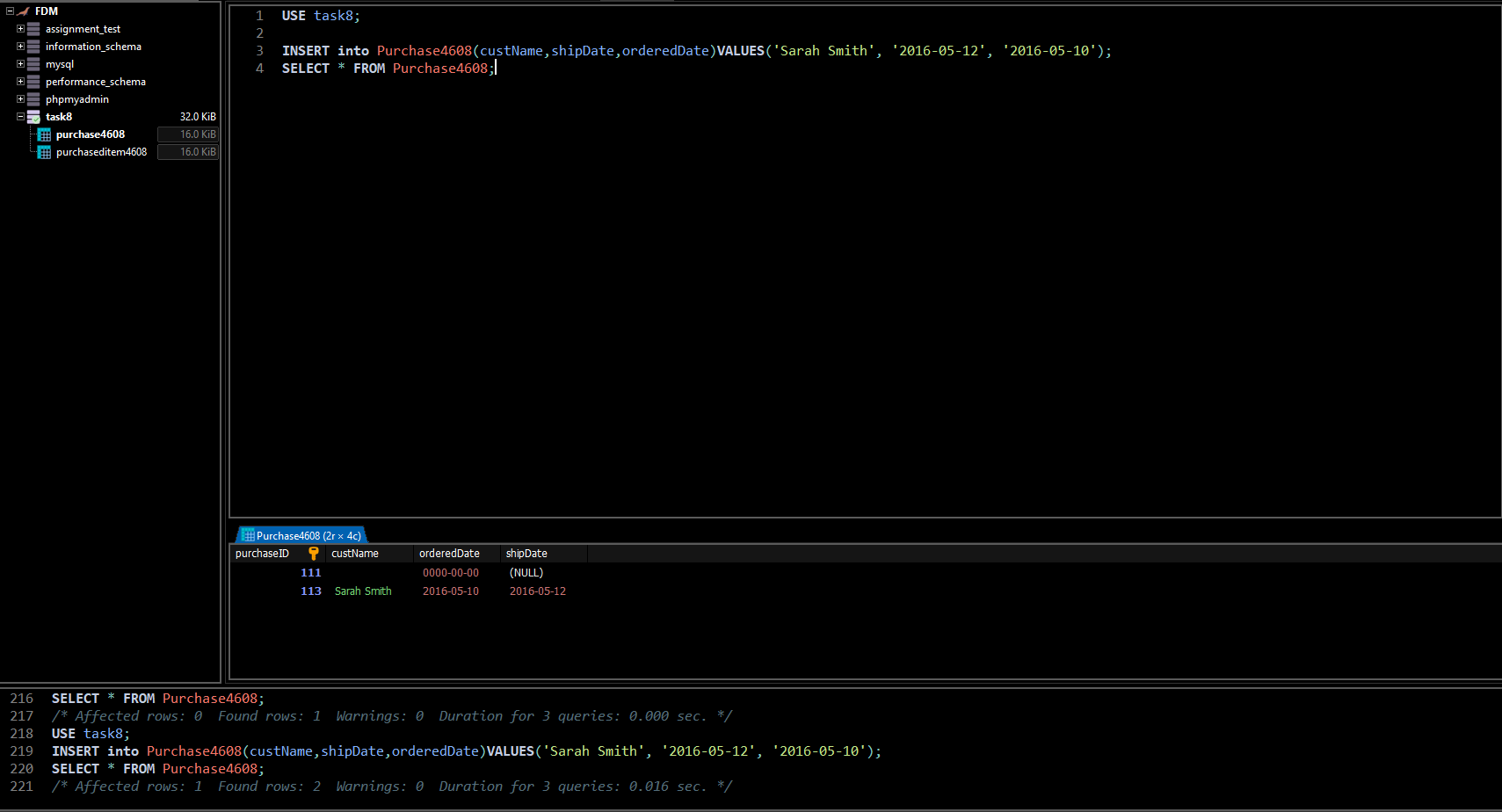
To fix this error we must fill the parent table first so the child table can connect to it. Then we use commit to save our changes.

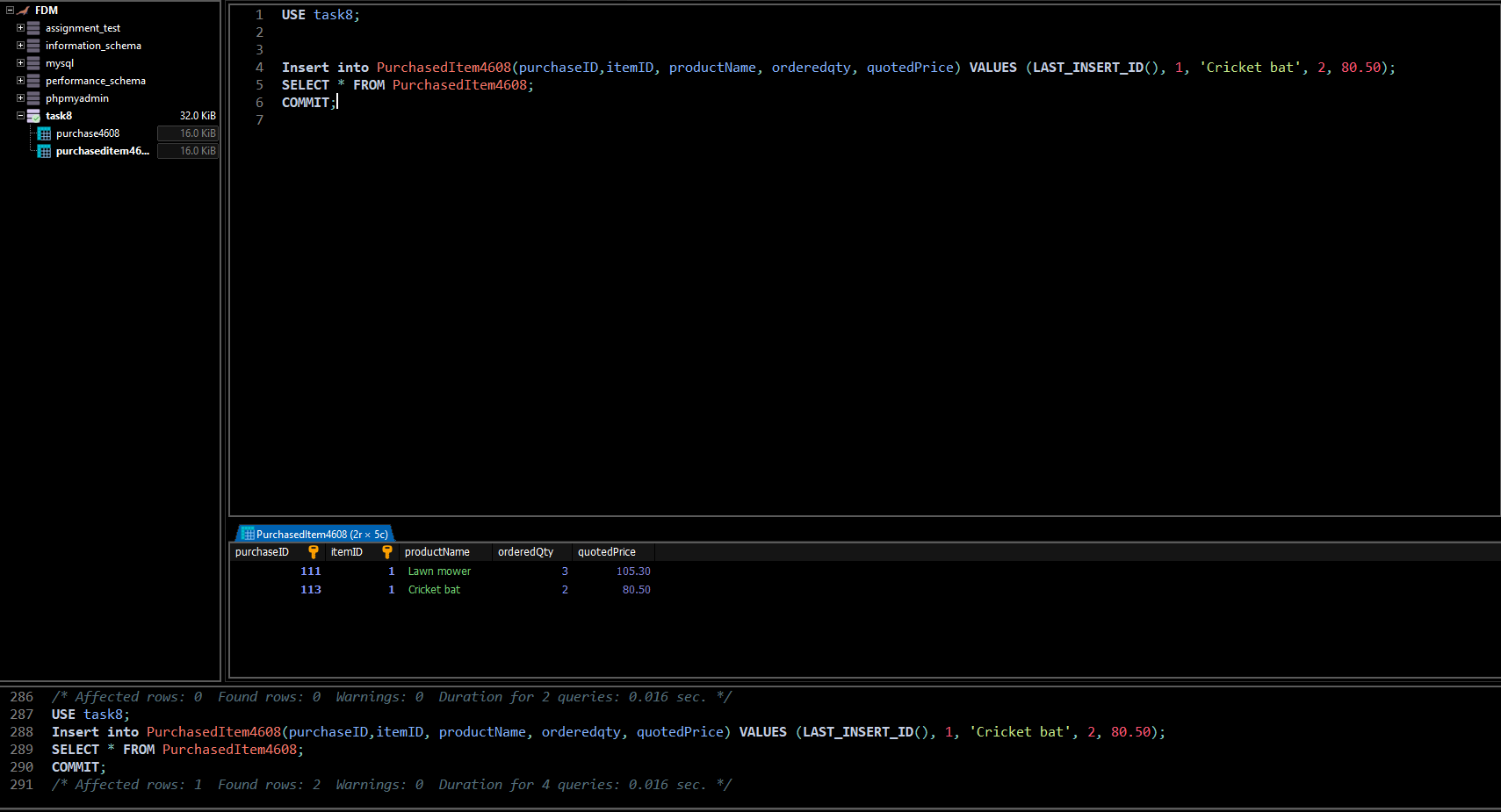
## SUB Task 8.1.2

#### Solution







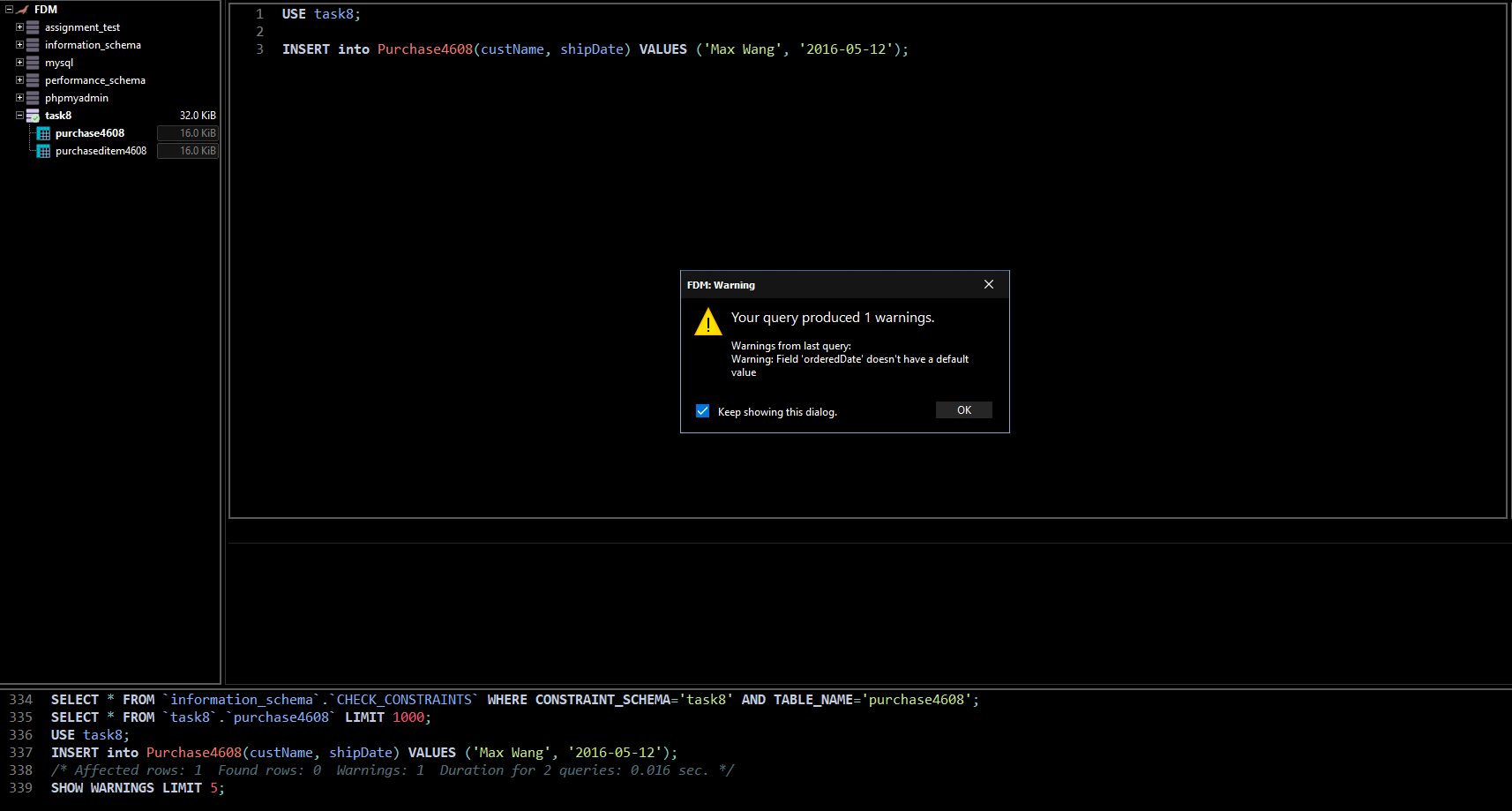


#### Comments/findings

The first screenshot and the second shows when a data was entered and removed from the table. In the first entry no commit was used so the data was not saved and in the second I used roll back, so it went to a state where I last committed, i.e task 8.1.1

## SUB Task 8.1.3

#### Solution





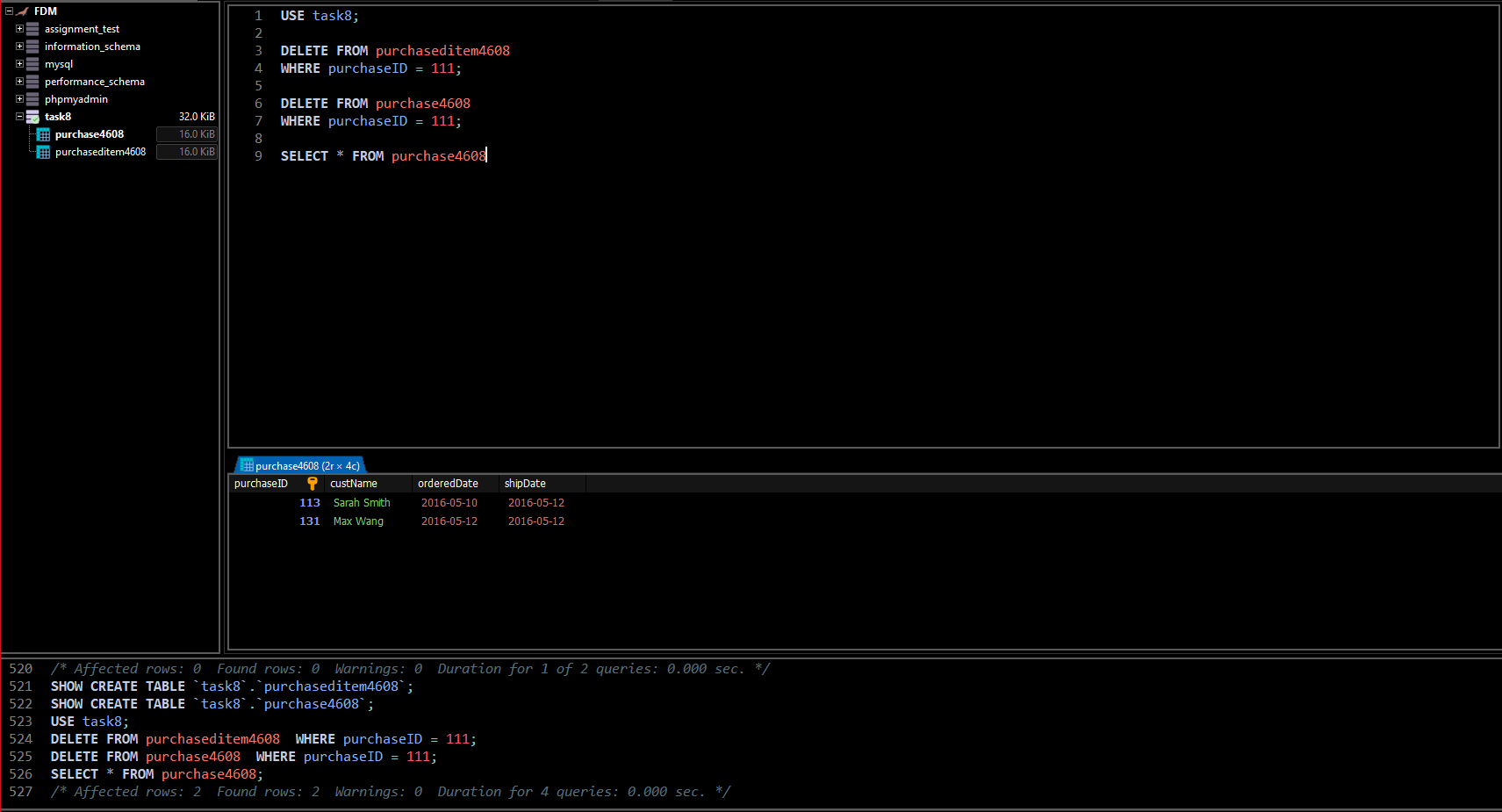
#### Comments/findings

It shows an error because orderDate is not provided and no default value for orderDate is set. To fix this I have to update the row in the table and provide a ordereDate so I chose the same date as shipDate for convenience.

## SUB Task 8.1.4

#### Solution

#### 



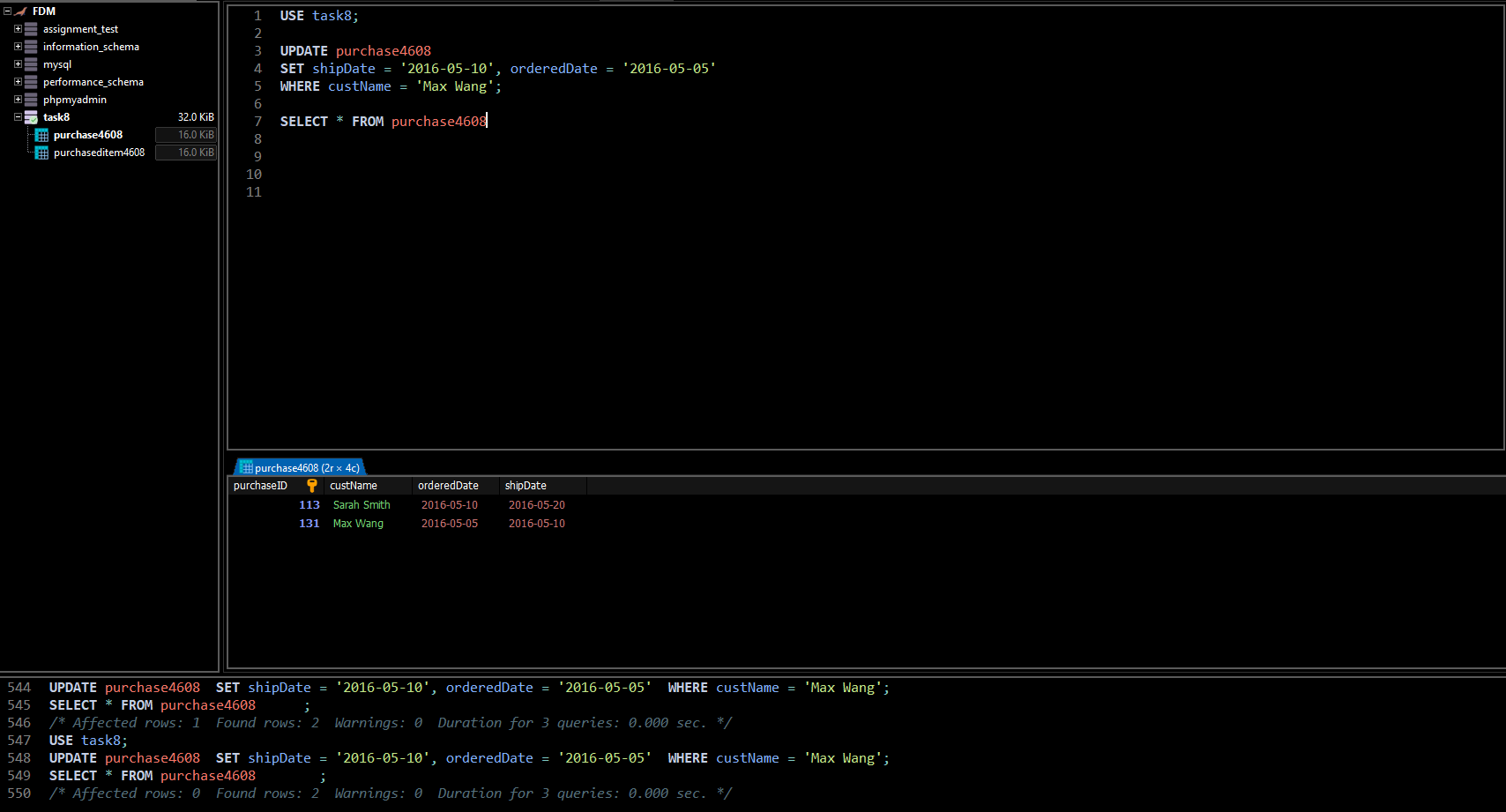
#### Comments/findings

Similar like task 8.1.1 we cannot delete a row from the parent table if it also exists in a child table, so we have to delete from child table first.

## SUB Task 8.1.5

#### Solution

#### 



#### Comments/findings

Just used update for both the task

Credit Task 8.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 8.2.1

#### Solution



#### Comments/findings

Everything can be select and interest at once using ( SELECT \* )

Pass Task 9.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 9.1.1

#### Solution

…

#### Comments/findings

This is because when a transaction happens three tables need to be updated and to do this we need to write three different statements.

Firstly, Products table was updated, since the amount of item ordered is 2 we need to reduce that quantity from the quantity we currently have thus “QuantityOnHand - 2”.

Next, Orders table was updated, it was written to identify the customer who placed the order, what items he got, when he placed the order and the shipping date.

Finally, Order\_Details table was updated it calculates the total/final price of the order that has been made.

Credit Task 9.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 9.2.1

#### Solution

**Run the first statement of T2 in your right MySQL Workbench instance. Run all statements of T1 in your left Workbench instance. What do you see?**



As we can see we obtained a result but there were no results for the 2nd and the 3rd statements after running in T1

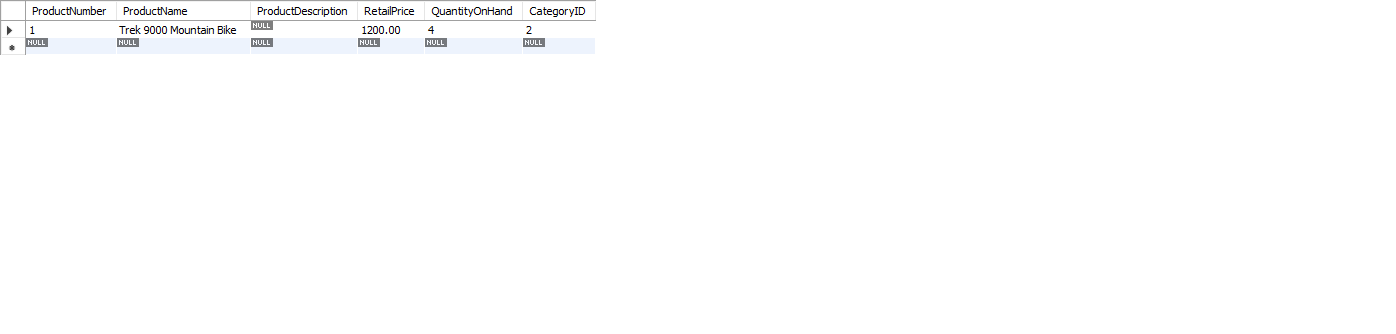
**Run the rest of T2 in the right MySQL Workbench. Check again what you can see in your left Workbench.**

Exactly the same as the previous question.

**Copy all statements of T1 into your right MySQL Workbench and run them. What do you see?**

Exactly the same as the previous question.

**Commit T2 in your right Workbench. Re-run T1 in your left instance. What do you see?**







**Commit T1 in your left Workbench. Re-run T1 again. What do you see?**

All the statements run properly as shown.

**What isolation level are you working at?**



**When does a transaction see the changes made? Answer for both the transaction that makes the changes and a transaction that merely reads the changes.**

T2 sees the changes made and T1 merely reads the changes, as we can see T1 changes only when the statements in T2 are run or committed and the only time we get to see a result is when we run T1.

**Why can’t T1 see the changes of T2 when T2 commits?**

Because the changes have not yet been saved.

**What do we mean by ‘repeatable read’ and do we have phantoms here in MySQL?**

Repeatable read makes sure that any data read cannot be changed until the reading is complete and won’t allow transaction to update the data.

Yes, we have phantoms in MySQL.

**What does the SQL standard say about phantoms and Repeatable Read Isolation level?**

A phantom read occurs when, during a transaction, new rows are added or removed by another transaction to the records being read. Repeatable read prevents phantom from having and is thus set as the default isolation level

#### Comments/findings

…

(Where applicable)

## SUB Task 9.2.2

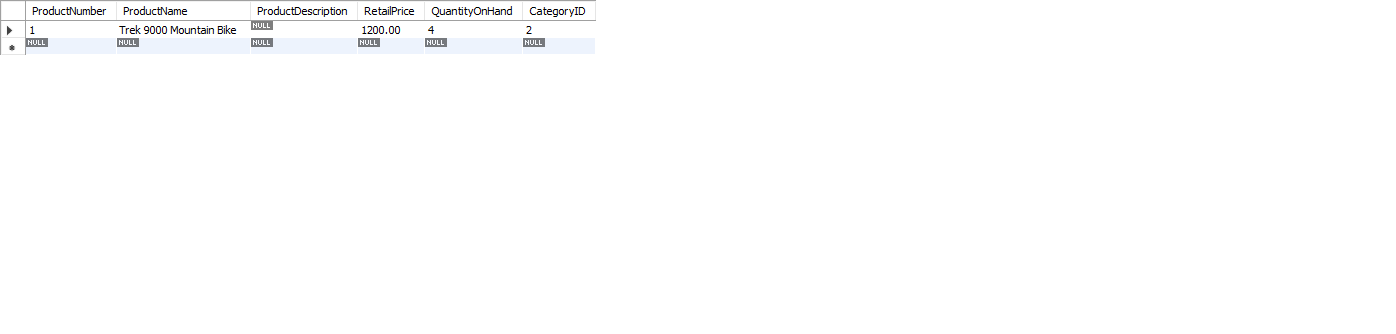
#### Solution

**Run the first statement of T2 again in your right MySQL Workbench instance. Run all statements of T1 in your left Workbench instance. What do you see?**



Just like before the first statement shows an output but the next two does not show any result.

**Run the rest of T2 in the right MySQL Workbench. Check again what you can see in your left Workbench.**







All the statements in T1 ran properly.

**Commit T2 in your right Workbench. Re-run T1 in your left instance. What do you see?**

All the statements run like before.

**Commit T1 in your left Workbench. Re-run T1 again. What do you see?**

Same as the previous task

**How did the query results differ from the ones in subtask 9.2.1? How can this difference lead to a lost update?**

The results are different because this time we could see all the changes even before we commit unlike previously where changes were only visible after committed.

We may think that the table was committed/updated not realizing they forgot to do so, which may lead to some results not being saved at all.

#### Comments/findings

…

(Where applicable)

## SUB Task 9.2.3

#### Solution

Read Committed isolation uses something called **cursor stability**, it is a locking strategy that locks the row the user has his/her cursor on. The row is locked until the transactions commits, once it is done it is automatically released from the row and positioned to the next row the cursor is on or if the cursor is closed. This prevents cursor lost updates that could occur if the locks are released immediate after data is read. This also allows for increased concurrency.

#### Comments/findings

…

(Where applicable)

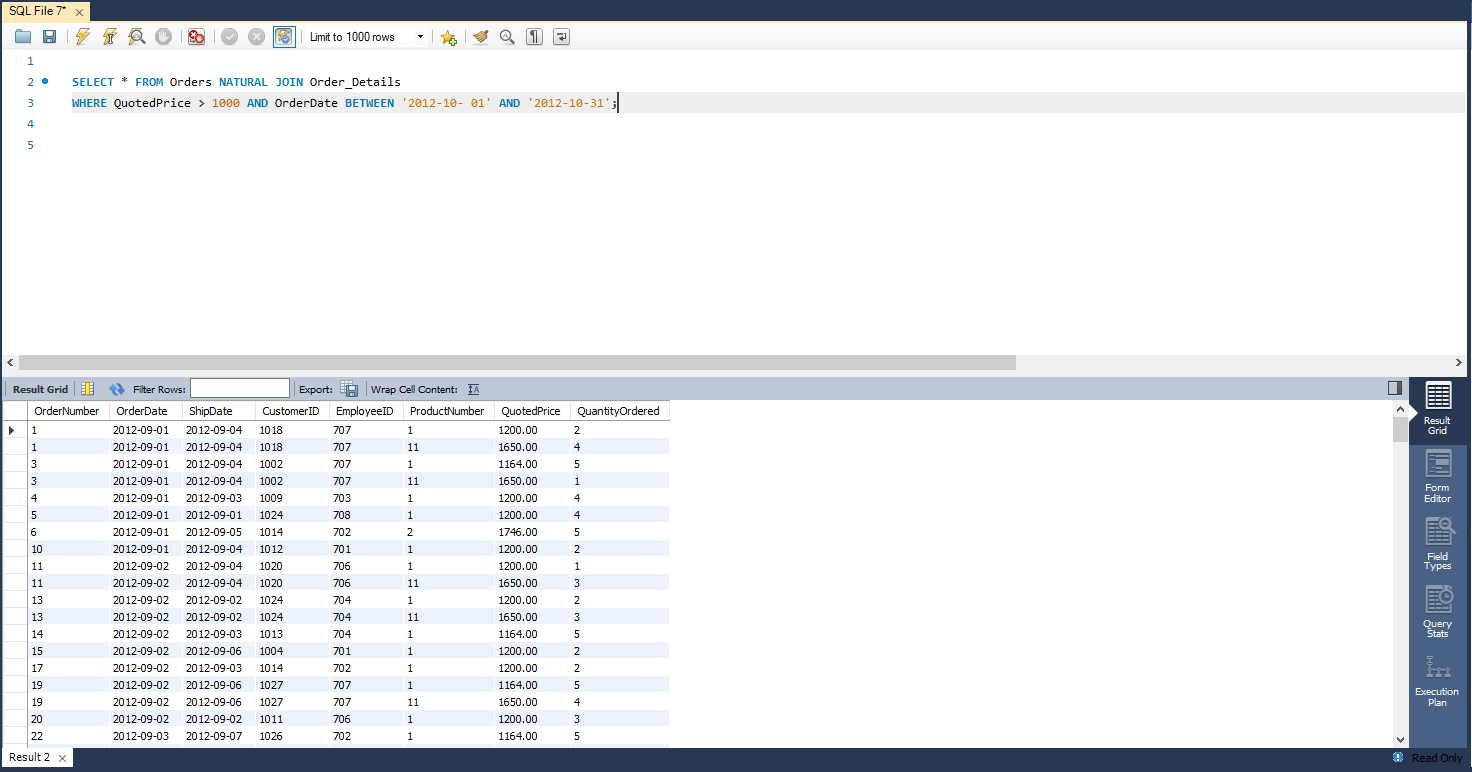
Credit Task 10.1 report

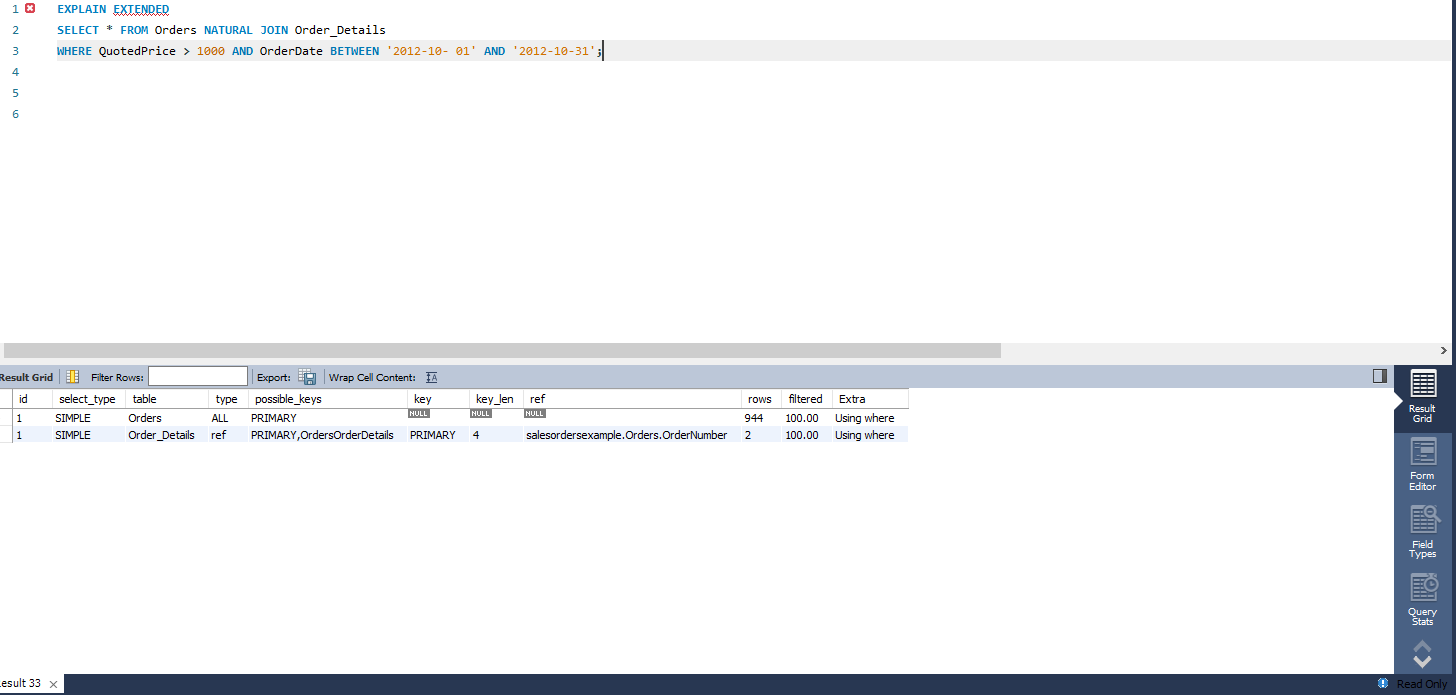
|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 10.1.1

#### Solution

**NO** this is not what I expected because there is an error in the code provided in the task, if I fix the error i,e the space between the date then **YES**.





From the output above it looks like the order table looked through the data rows and the order\_details table used and index. Order table was accessed first because it’s a parent table

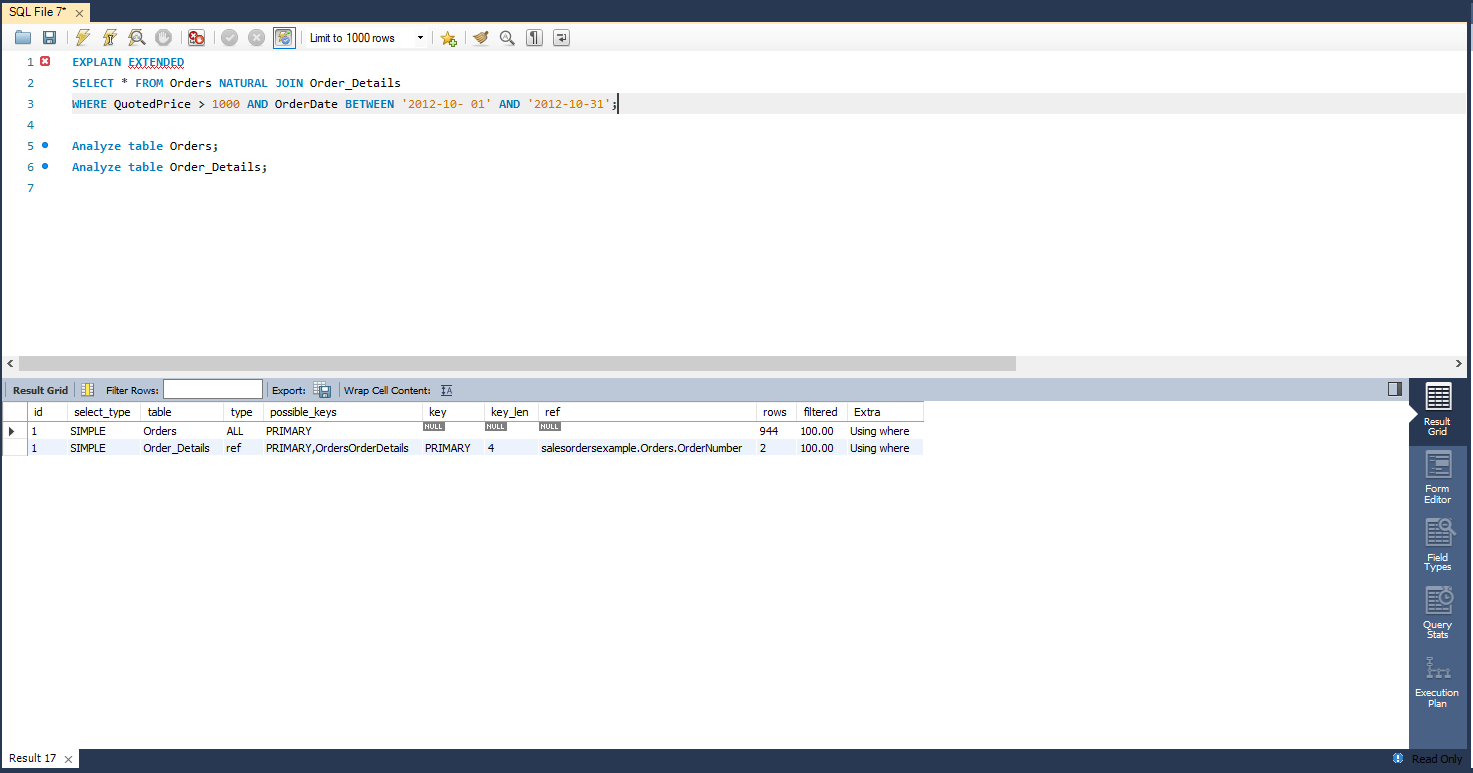
#### Comments/findings

…

(Where applicable)

## SUB Task 10.1.2

#### Solution



Nothing changes at all for me, I tried recreating the database several times, no change. But apparently the number of rows is supposed to increase by 1 from 944 to 945.

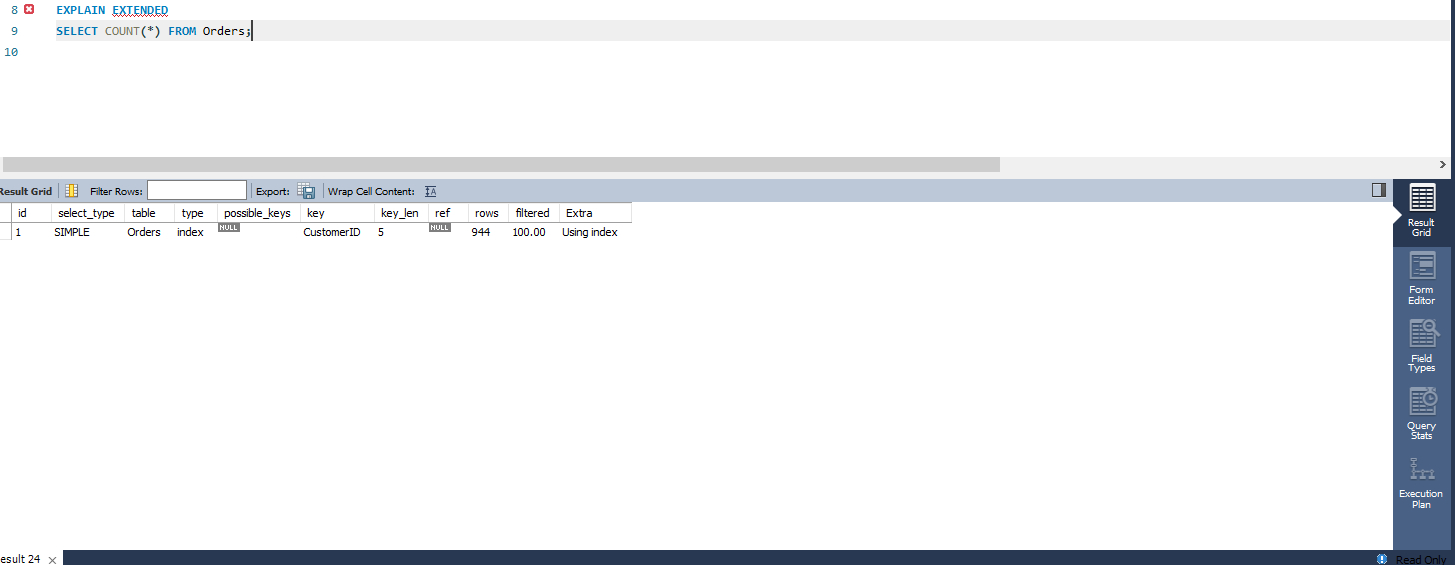
#### Comments/findings

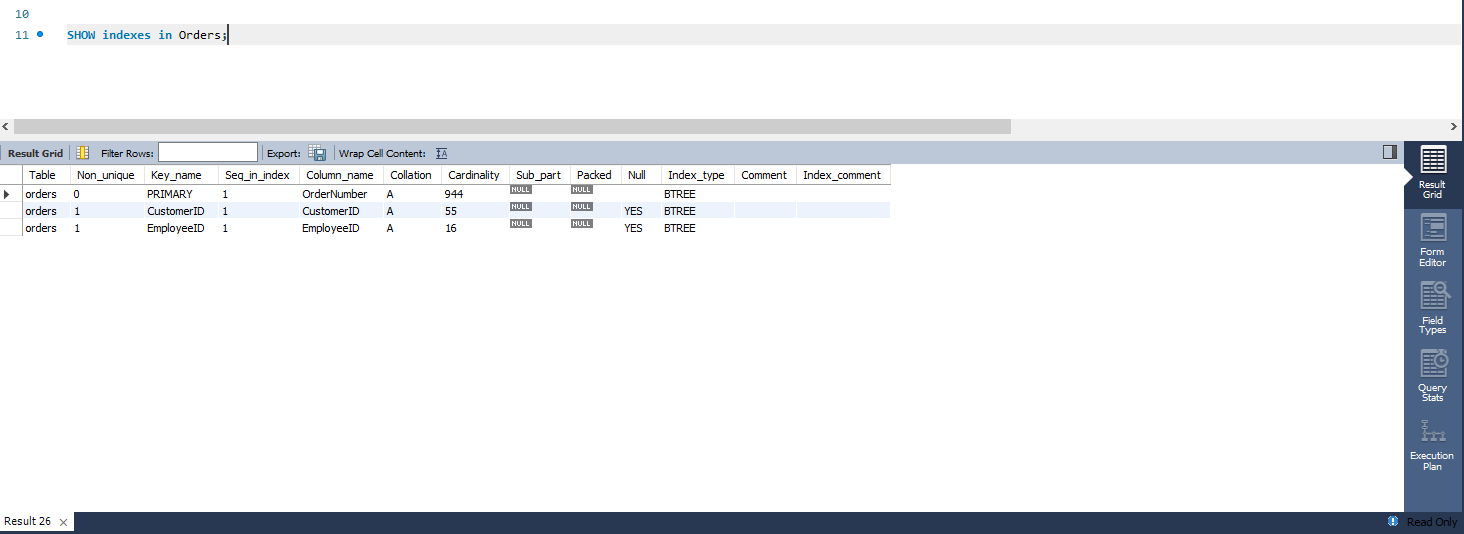
I decided to continue the task, I have even tried to download the script and running again, nothing changed.

## SUB Task 10.1.3

#### Solution





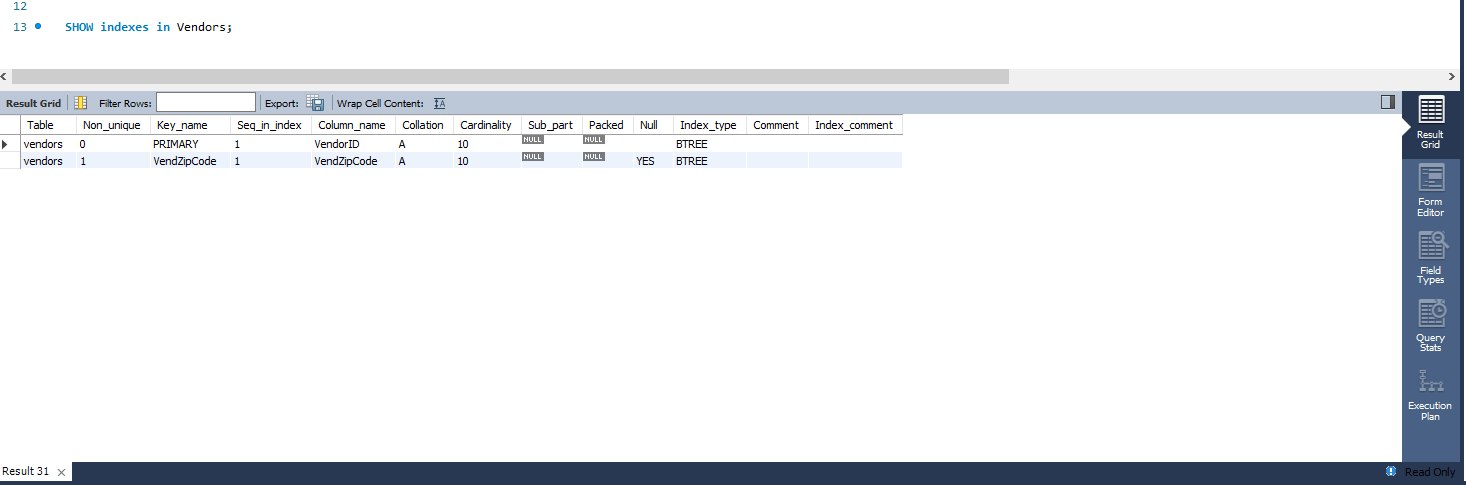


#### Comments/findings

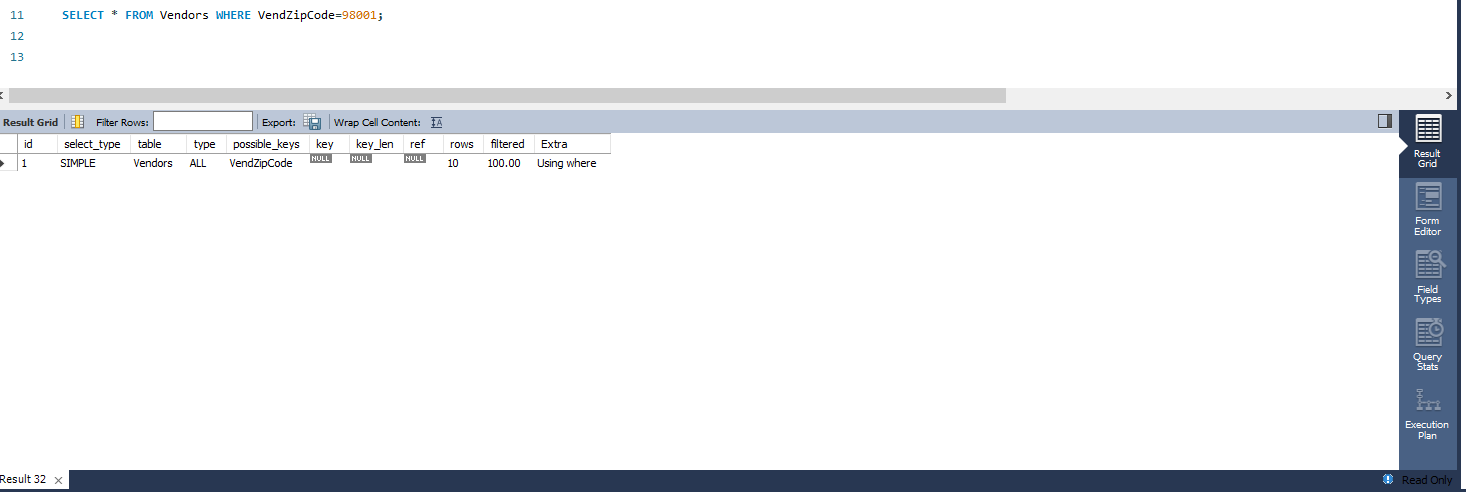
We can see that dbms used 3 indices find the total number of rows in the table, which shows that dmbs used these to more efficiently and easily go through the data rows individually.

## SUB Task 10.1.4

#### Solution







#### Comments/findings

We can see that no reference was used as it went through each row one by one.

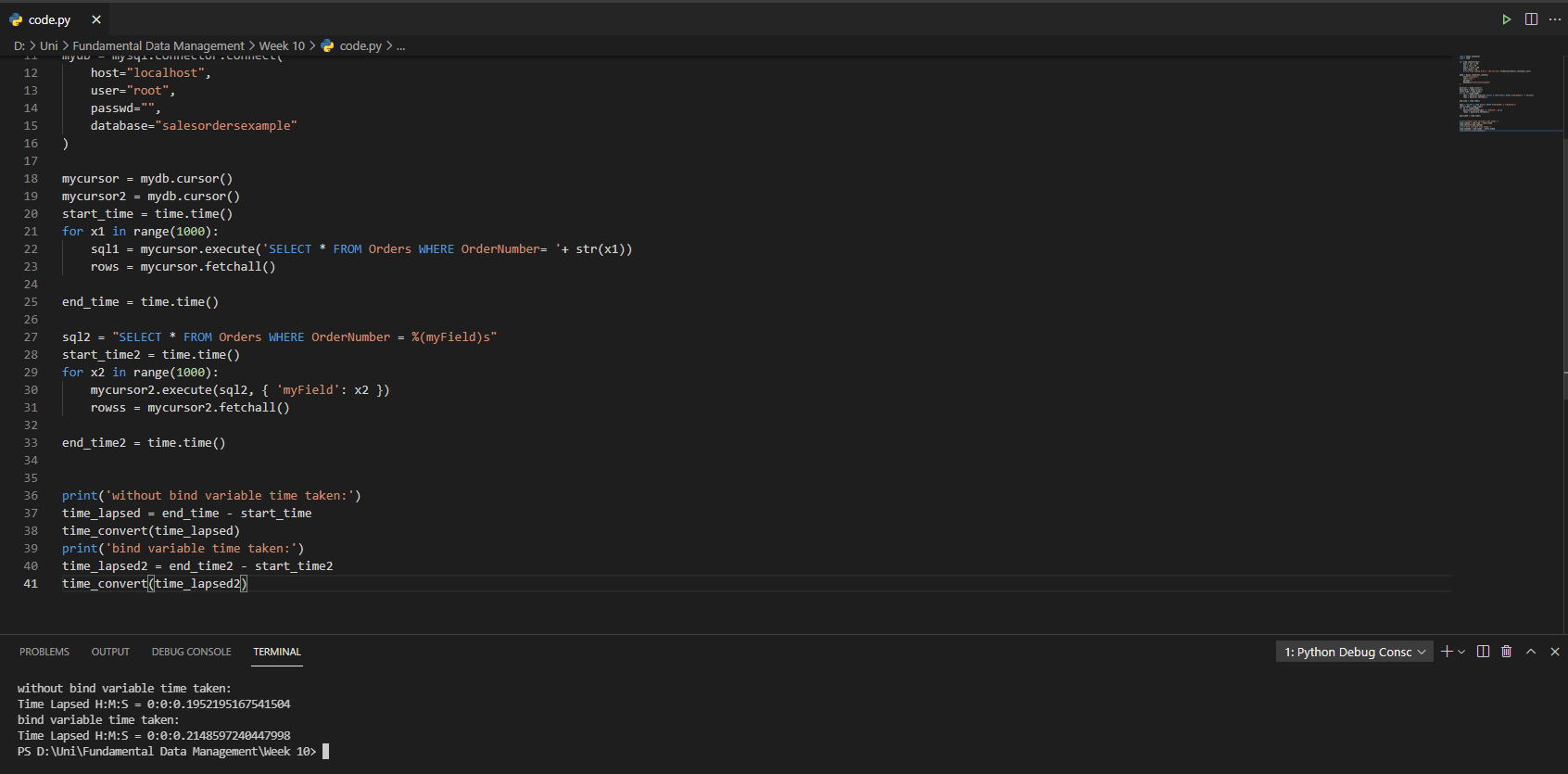
YES, it was executed the way I expected.

Distinction Task 10.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 10.2.1

#### Solution



#### Comments/findings

From the screenshot above it is evident that a search query with bind variable is faster than a query without but the difference is very small almost insignificant, I ran the program multiple times and some time the difference in result were even smaller.

It takes a longer time because before SQL runs the query it checks if its valid and determines how to access the tables. The optimizer has the task of figuring out which table access to use, what to with null values. This produces an execution plan thus taking a longer time to execute.

Credit Task 11.1 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 11.1.1

#### Solution

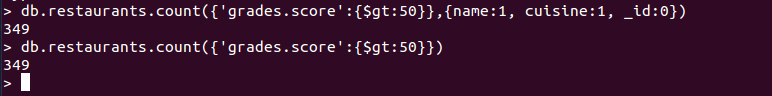
1. db.restaurants.find({'grades.score':{$gt:50}},{name:1, cuisine:1, \_id:0})



1. db.restaurants.count({'grades.score':{$gt:50}},{name:1, cuisine:1, \_id:0})

or

db.restaurants.count({'grades.score':{$gt:50}})



#### Comments/findings

For part b) I tried both the commands, not sure which one is require and they both gave same result so I showed them both.

## SUB Task 11.1.2

#### Solution

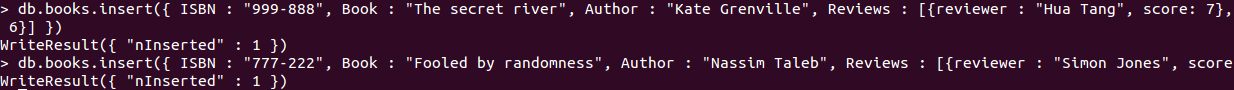
1. db.createCollection(‘books’)



1. db.books.insert({ ISBN : "999-888", Book : "The secret river", Author : "Kate Grenville", Reviews : [{reviewer : "Hua Tang", score: 7},{reviewer : "Jim Wong", score: 8},{reviewer : "Andrea Bauer", score: 6}] })

and

db.books.insert({ ISBN : "777-222", Book : "Fooled by randomness", Author : "Nassim Taleb", Reviews : [{reviewer : "Simon Jones", score: 5},{reviewer : "John Messum", score: 10}] })



#### Comments/findings

The insert code is very long and when the screen shot was taken its difficult to read the values so a shorter screen shot is shown but the code used to pasted above

## SUB Task 11.1.3

#### Solution

1. db.books.update({Author: "Kate Grenville"},{$set: {Author: "Simon Collins"}})



When the update is successful it changes the nModified number, and since I only changed 1 value it says .

1. Upsert is a combination of update and insert, it inserts or updates an existing record in one call. Basically, if the row does not exist it inserts into the database and if they do exists then they update the row.

#### Comments/findings

…

(Where applicable)

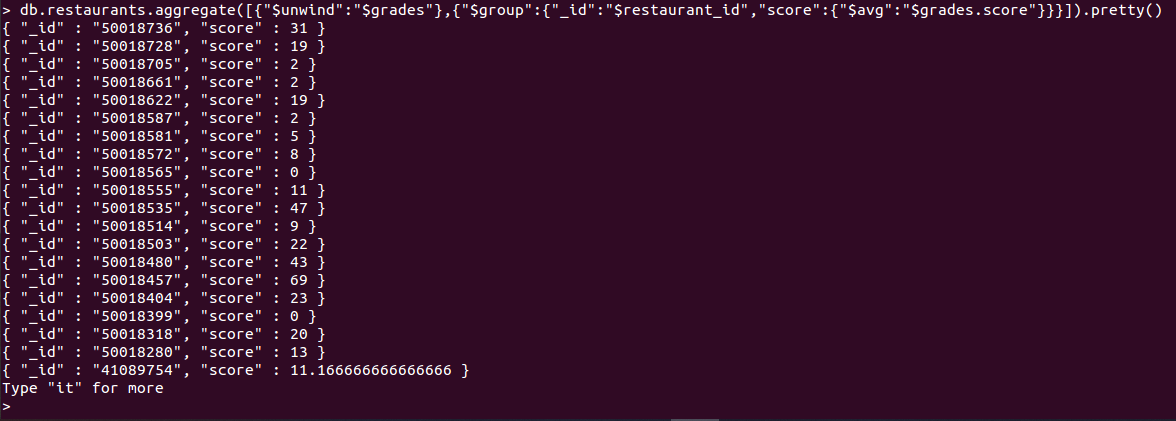
Distinction Task 11.2 report

|  |  |
| --- | --- |
| Student Name | **Masrur Rahman Zahin** |
| Student ID | **101214608** |

## SUB Task 11.2.1

#### Solution

db.restaurants.aggregate([{"$unwind":"$grades"},{"$group":{"\_id":"$restaurant\_id","score":{"$avg":"$grades.score"}}}]).pretty()



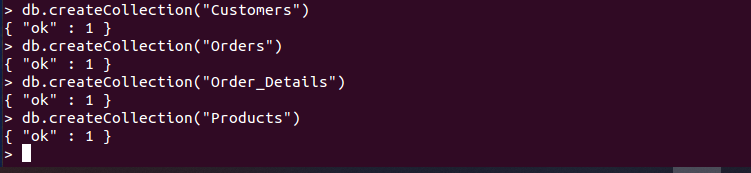
#### Comments/findings

Too a while to get it right, but eventually got it. **.pretty()** was used just to make it look better and easy to read.

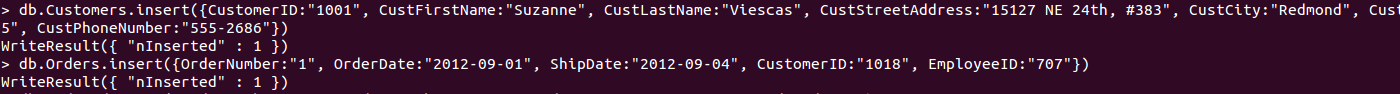
## SUB Task 11.2.2

#### Solution

1. db.createCollection(“<CollectionName>”)



1. db.<CollectionName>.insert{( )}







#### Comments/findings

1. I chose an easy and safe method to create my collections, since my collection needed is small I used 4 lines of code.
2. For part b) I just inserted the first data provide in the week 10 VM folder, I only inserted one data as I am just trying to show how it works.

# Conclusion

By completing this unit I was able to learn how to design and manage a database and how a database works. I completed all my tasks on time including my D report tasks and I believe I should be able to obtain at least a D grade.