**CPSC 50900 Database Systems Project**

All your efforts this semester will focus on this project to be done BY YOURSELF for which you will develop, document, implement, secure, and program with a database. You will post the **artifacts of work** (XML files, model diagrams, code) as a **GitHub repository**, and you will add to it steadily throughout the semester. You will document all your efforts in a single **Word document** that you'll write throughout the semester, sharing it with me periodically to review and provide comment.  
  
**To summarize: Create a report using Word, and store all of your writing there. For technical documents, post those to a GitHub account and include the link in your report at the end of your proposal.**   
  
The report must have a title page, and the total is worth 10 points for clarity and organization.  
Organizing your files in GitHub by type with clear and unique file names is worth another 10 points.

Your project will consist of the following sections:

**Initial Proposal**

In your report document:

You will describe the data you aim to store. What is your application or business? What data will be storing? What is the value of this data? Where will the data come from? Who will use this data?

Rubric: Your response to each of these five questions will be graded out of 3 points.

* 3 points: clear, complete descriptions that convey the importance and meaning of your data
* 2 points: mostly clear descriptions, although some additional data would have helped in some sections – If it sounds like you are writing without saying anything (“fluff”), then you are losing a point.
* 1 point: necessary details are lacking in many of your responses.

Be aware that this is not the final summary of what you will create in your summary. This is just an overview.

For example, my application is to sell robots. I will be storing business data and customer data. The value of business data helps me track and improve my business, the value of customer data helps me maintain relationships. I get supply chain data from my suppliers, I create the parts data myself when I create the products for my catalog, I get customer data when customers place orders. I use all of the data, my employees will use the inventory and sales data, and customers will access inventory in the form of a catalog. My writing might look like this:  
  
“I have created a business selling robotic systems. I need to create a database solution that keeps track of customer data, product data, employee data, store data and supply chain data. I’m interested in this data because I need to track my business operations to make sure that I have the right components, I need to watch inventory and order more materials if I run low, I need to maintain customer records so that I can build relationships with my customers and offer discounts for customer loyalty, and I need to track my employees so that I can keep my stores staffed and employees paid. I will manually enter employee data, but other data comes from my suppliers, or customers, or whatever, through a server process (Salesforce, SAP, Oracle, using middleware?).

Files are located at:  
[www.github/mattsRobots](http://www.github/mattsRobots) (not a real address)

Your report might look similar, but please don’t copy what I wrote.

Total points possible: 15

I have created a business selling motorcycle parts. I will store customer and business data in it. The data helps me track and improve business, customer data helps me improve relationtionship with customer and keep track of their purchases. I need to keep track of inventory so that when I run low I can order more.

I will also keep track of my suppliers so that when I need to contact them I can do so easily. Also, I will keep track of purchases.

Files are located at:

**Relational Database Design Process**

Identify at least 5 entity sets and their attributes (at least 3 attributes per entity), and describe these entities in your **report**. This will look a lot like the description component of the Aquarium assignment. Then illustrate that you understand normalization and how to use it to reduce uncontrolled redundancy in your database design by creating a single ERD diagram using either UML or Crow’s Foot notation and upload that drawing to **GitHub**. The ERD must be a **physical** model (includes data types and key information).

Rubric: Your work will be graded as follows:

* 4 points for describing each of 5 entities and their attributes in your report
* 10 points for the physical model in GitHub

Total points possible: 25

employees (First\_name, Last\_name, Phone\_number, Hired\_date,)

Order (Order\_quantity, Order\_date\_time, Phone\_number, Hired\_date,)

product (Name, Description, Quantity\_stock, Price,)

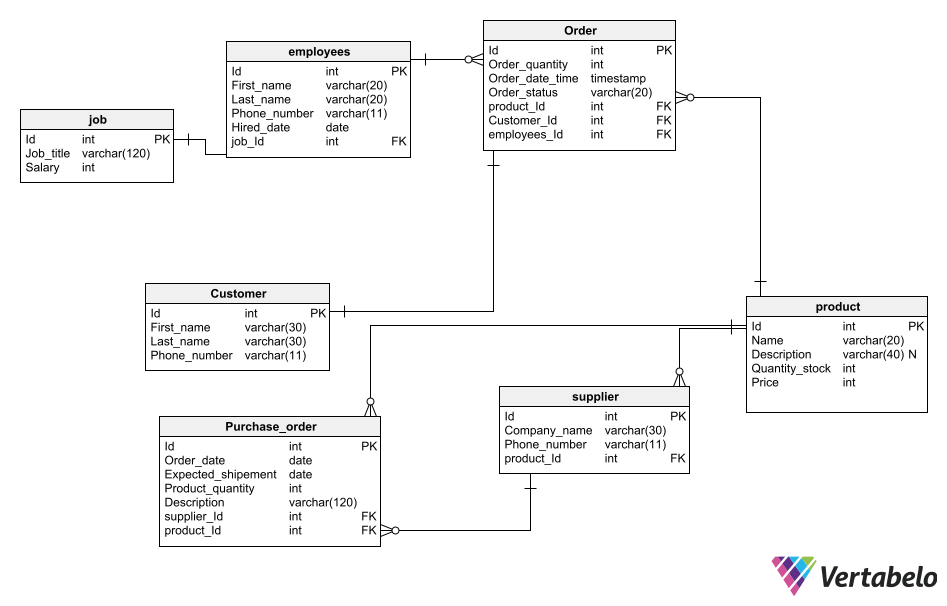
supplier (Company\_name, Phone\_number,)

Customer (First\_name, Last\_name, Phone\_number,)

Purchase\_order (Order\_date, Expected\_shipement, Product\_quantity, Description)

Purchase\_order (Job\_title, Salary)

Physical model is also uploaded on github:



**Data Sources**

Create examples (**three** records per entity) of all of your data in a parseable format such as XML, JSON, or delimited like comma separated values and submit the files to **GitHub**. In your **report**, explain the process of how your software stores each entity and attribute, what is the source of the attributes, and what data is default.  You don’t have to describe every attribute, but I want to know at least two details about each entity, and how that data enters the system.  
  
My example would include something like “As the business owner I would manually input my employee data (entity) which includes name, phone number, and address (specific details and attributes) into the database. The database updates the employee ID automatically (a default condition that defines the remaining attribute of my entity). However, I use an arglebargle software tool (and if I see arglebargle in your report, you are losing a lot of points!) to read product information (another entity) and format the part number, description, cost, and type (all the attributes of the product entity) into an XML data structure. All of these attributes come from the supplier. ” Notice, I was specific with these attributes and determined which were default, which were created, and where they all came from.

Rubric: Your work will be graded as follows:

* 5 points per populated file that has attributes which **match your design.** If you are missing one or two, then you lose a couple points. If you miss a lot, you may lose half or all the points.
* 5 points per entity: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.

Total points possible: 50

I have uploaded the files xls format to github

As owner I will personally add customers, employees, job, product, purchase\_order, and supplier information in database. Other employees will be able to view and edit orders. All ids will be auto incremented. Products are viewable by other employees as well. **Data Definition Language Scripts**

OLD INSTRUCTIONS:

First, use Vertabello to generate a script of SQL commands that build the database and its table structures. Or, you can write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL *source* command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.

UPDATED INSTRUCTIONS:

To simplify the instructions, first you need a script that creates the table structures.

Then, you need the ‘load’ command to import the data you saved to an XML or CSV file. You might have also used JSON or another format, but you’ll have to research that method yourself.

Assign columns as a **Primary key** or **Foreign key** (where needed) so that if you modify a row in one table, the changes update in the other tables. Obviously, do not assign columns as keys if they are not shared across tables.

In both of the following sections, I offer you CHOICES (by number)– not steps to complete. Pick your favorite ONE, and execute that choice.

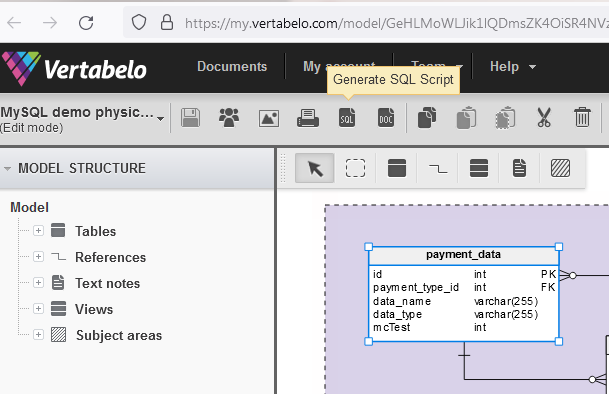
TO CREATE THE TABLE:

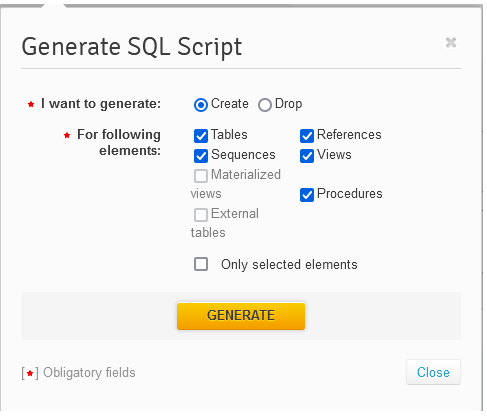
1. Manually write a script with CREATE and VALUE statements. You can either write this at the prompt (can be tedious if you make errors), copy/paste a script into the prompt, or save your script as a .sql file and use ‘SOURCE’ to load your script like you did when you loaded the nation database for the SQL#2 assignment.

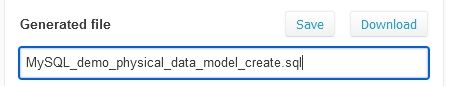
You may feel this option is easier/faster than using Vertebelo. However, if you drew your model in Vertebelo, then the script is just a 4 step point-and-click process.

2. Generate the script using a tool like Vertebelo. Here is the procedure:

a) Create the tables/database:



b) Using the SQL icon above the drawing area, choose to Generate SQL Script: 

c) Click on ‘Generate’ to export the SQL file:  


d) download, and copy the .sql file to your favorite directory where you administrate MariaDB. This is where you use the ‘SOURCE’ command to load the sql file like you did with the nation db example.

TO IMPORT DATA INTO MariaDB:

1. Use “LOAD DATA LOCAL INFILE ‘filename’ “ to load a delimited Excel file (usually a comma separated value, CSV).

<https://mariadb.com/kb/en/load-data-infile/>

You’ll notice the ‘set’ at the end gives you the option to manipulate data during the load. In fact, nearly all of those commands are optional since they are shown within square brackets, [ ]. Thus, this step is very simple! You may also redefine the delimiter character if you are not using a CSV.

2. If you are using XML files to store data, then see this example:

<https://mariadb.com/kb/en/load-xml/>

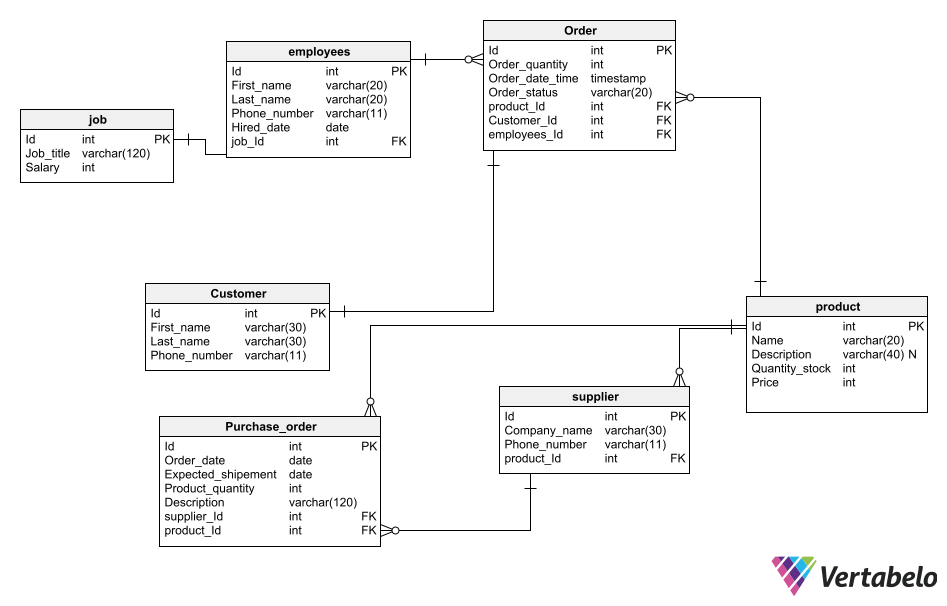
Again, a single command will load the contents of your file.

Rubric: Your work will be graded as follows:

* Database and table creation statements (manual or Vertebelo) saved as an sql script file. Upload to GitHub. 8 points
* The SQL commands for populating the tables **and** what they do (explain each of the commands and options that you used in the script) documented in your report: 8 points
* Screenshots of your successful attempts to populate each table with at least three records: 4 points

Total points possible: 20

Description of entities and their attributes



Physical design:  
CREATE TABLE Customer (

Id int NOT NULL,

First\_name varchar(30) NOT NULL,

Last\_name varchar(30) NOT NULL,

Phone\_number varchar(11) NOT NULL,

CONSTRAINT Customer\_pk PRIMARY KEY (Id)

);

-- Table: Order

CREATE TABLE `Order` (

Id int NOT NULL,

Order\_quantity int NOT NULL,

Order\_date\_time timestamp NOT NULL,

Order\_status varchar(20) NOT NULL,

product\_Id int NOT NULL,

Customer\_Id int NOT NULL,

employees\_Id int NOT NULL,

CONSTRAINT Order\_pk PRIMARY KEY (Id)

);

-- Table: Purchase\_order

CREATE TABLE Purchase\_order (

Id int NOT NULL,

Order\_date date NOT NULL,

Expected\_shipement date NOT NULL,

Product\_quantity int NOT NULL,

Description varchar(120) NOT NULL,

supplier\_Id int NOT NULL,

product\_Id int NOT NULL,

CONSTRAINT Purchase\_order\_pk PRIMARY KEY (Id)

);

-- Table: employees

CREATE TABLE employees (

Id int NOT NULL,

First\_name varchar(20) NOT NULL,

Last\_name varchar(20) NOT NULL,

Phone\_number varchar(11) NOT NULL,

Hired\_date date NOT NULL,

job\_Id int NOT NULL,

CONSTRAINT employees\_pk PRIMARY KEY (Id)

);

-- Table: job

CREATE TABLE job (

Id int NOT NULL,

Job\_title varchar(120) NOT NULL,

Salary int NOT NULL,

CONSTRAINT job\_pk PRIMARY KEY (Id)

);

-- Table: product

CREATE TABLE product (

Id int NOT NULL,

Name varchar(20) NOT NULL,

Description varchar(40) NULL,

Quantity\_stock int NOT NULL,

Price int NOT NULL,

CONSTRAINT product\_pk PRIMARY KEY (Id)

);

-- Table: supplier

CREATE TABLE supplier (

Id int NOT NULL,

Company\_name varchar(30) NOT NULL,

Phone\_number varchar(11) NOT NULL,

product\_Id int NOT NULL,

CONSTRAINT supplier\_pk PRIMARY KEY (Id)

);

-- foreign keys

-- Reference: Order\_Customer (table: Order)

ALTER TABLE `Order` ADD CONSTRAINT Order\_Customer FOREIGN KEY Order\_Customer (Customer\_Id)

REFERENCES Customer (Id);

-- Reference: Order\_employees (table: Order)

ALTER TABLE `Order` ADD CONSTRAINT Order\_employees FOREIGN KEY Order\_employees (employees\_Id)

REFERENCES employees (Id);

-- Reference: Order\_product (table: Order)

ALTER TABLE `Order` ADD CONSTRAINT Order\_product FOREIGN KEY Order\_product (product\_Id)

REFERENCES product (Id);

-- Reference: Purchase\_order\_product (table: Purchase\_order)

ALTER TABLE Purchase\_order ADD CONSTRAINT Purchase\_order\_product FOREIGN KEY Purchase\_order\_product (product\_Id)

REFERENCES product (Id);

-- Reference: Purchase\_order\_supplier (table: Purchase\_order)

ALTER TABLE Purchase\_order ADD CONSTRAINT Purchase\_order\_supplier FOREIGN KEY Purchase\_order\_supplier (supplier\_Id)

REFERENCES supplier (Id);

-- Reference: employees\_job (table: employees)

ALTER TABLE employees ADD CONSTRAINT employees\_job FOREIGN KEY employees\_job (job\_Id)

REFERENCES job (Id);

-- Reference: supplier\_product (table: supplier)

ALTER TABLE supplier ADD CONSTRAINT supplier\_product FOREIGN KEY supplier\_product (product\_Id)

REFERENCES product (Id);

**Data Manipulation Language Scripts**

Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document **report**, and save your queries in a commented sql script to GitHub. Points for the SQL statements will be scored in the scripts on GitHub, points for the screenshots will be scored in your report.

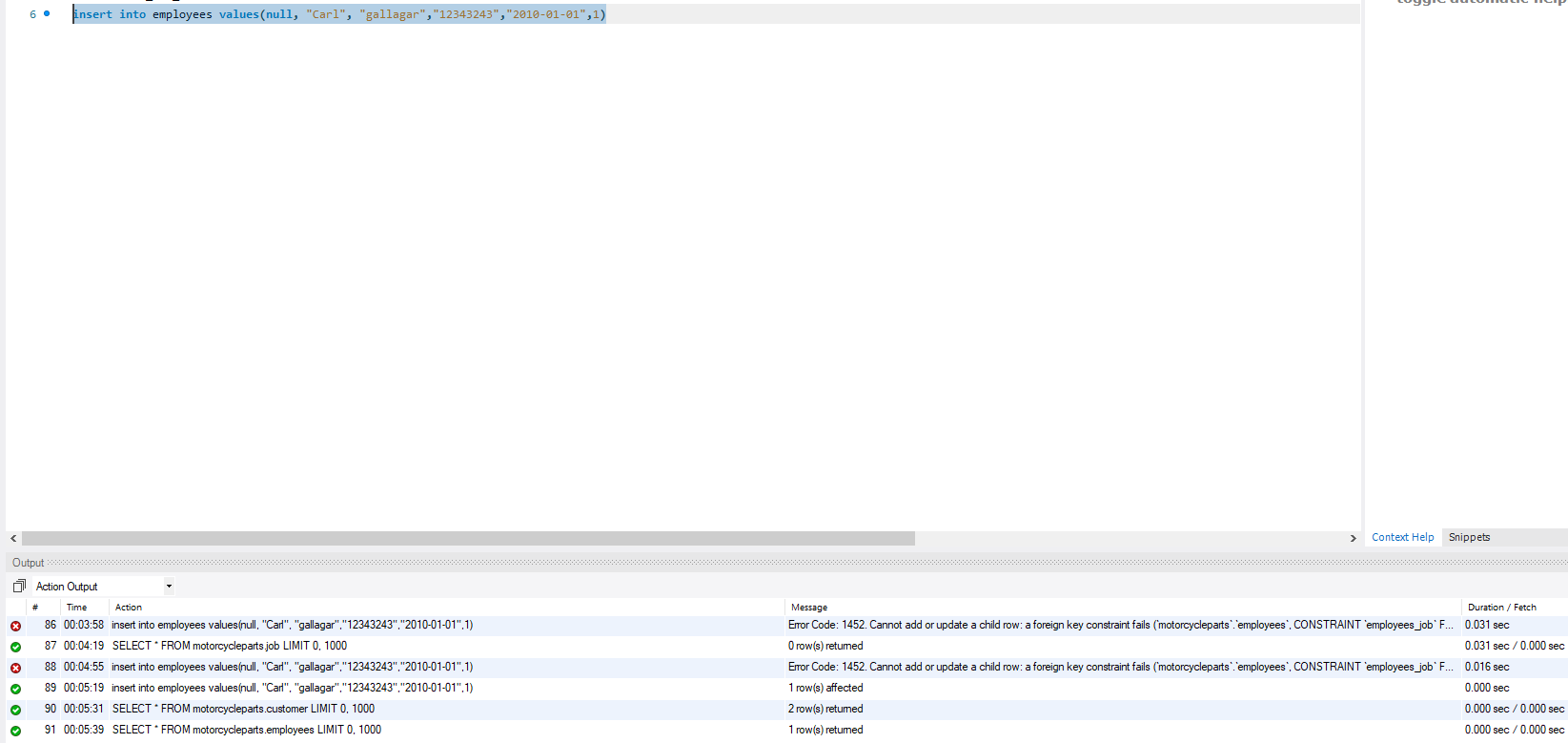
Rubric: Your work will be graded as follows:

* 1 point each for the two insert statements
* 1 point each for the two update statements
* 1 point for the delete statement
* 1 point for the simple select statement
* 2 points each for the 2 join statements
* 2 points each for the two that use summary statements
* 2 points for the multi-table query
* 2 points for the query of your choice.
* 12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.

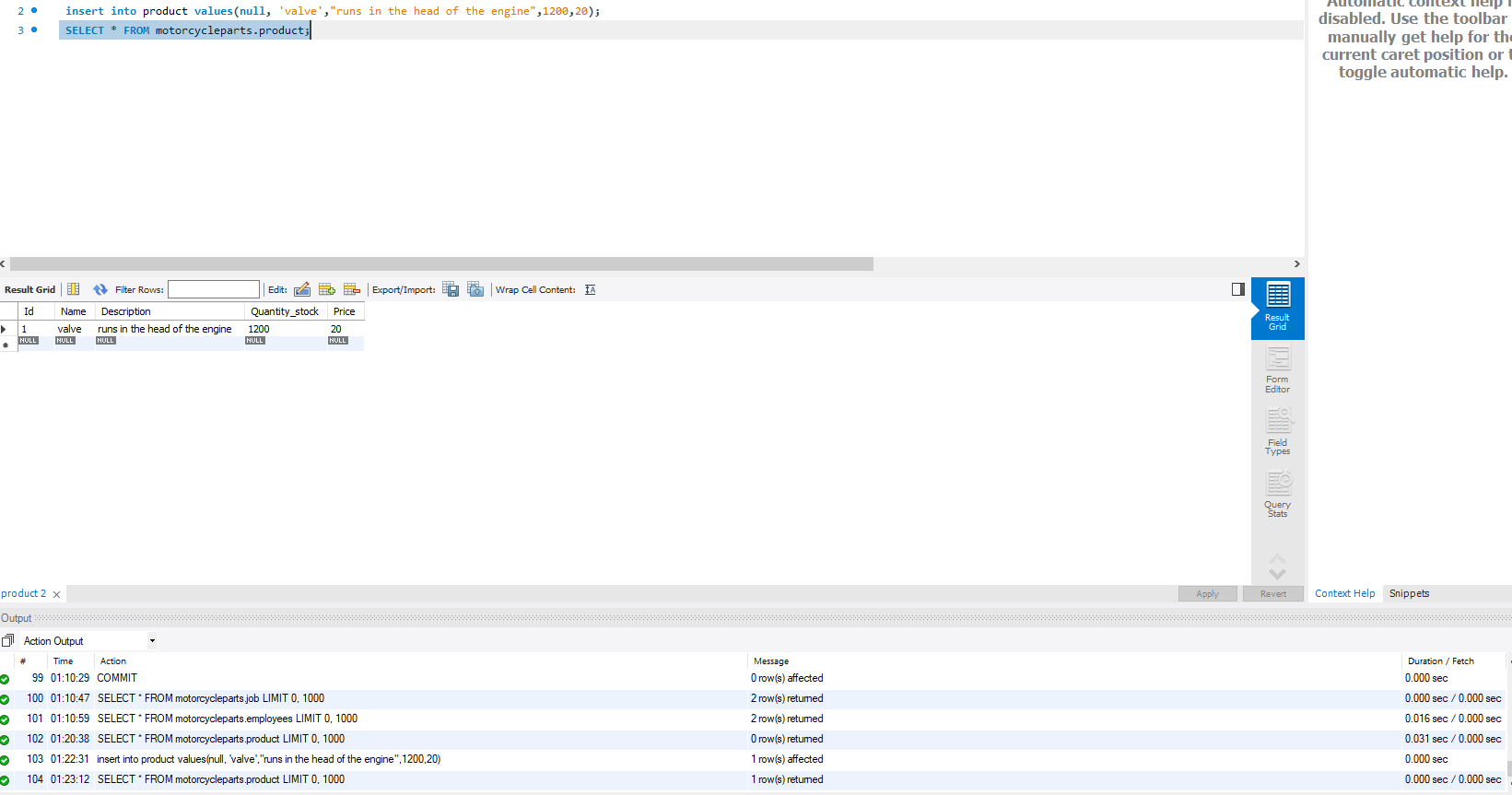
Total points possible: 30

**INSERT STATEMENT**

insert into employees values(null, "Carl", "gallagar","12343243","2010-01-01",1)

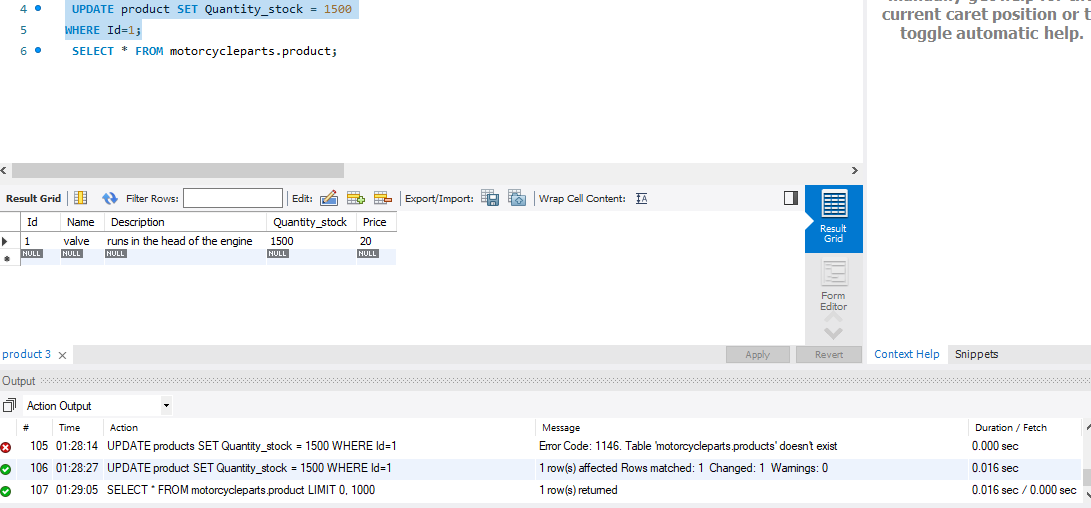


insert into product values(null, 'valve',"runs in the head of the engine",1200,20)



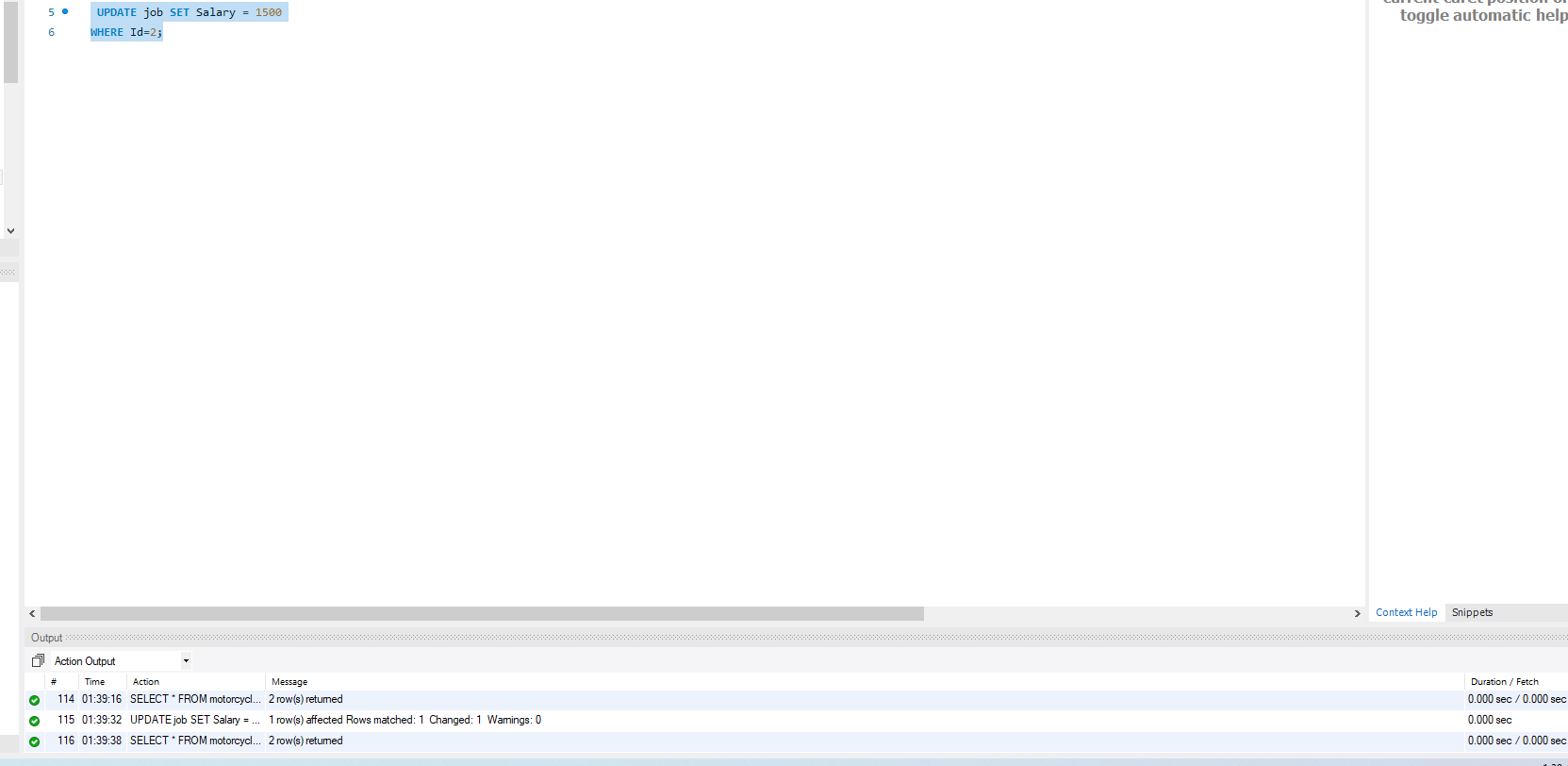
**UPDATE COMMAND:**

UPDATE product SET Quantity\_stock = 1500

WHERE Id=1;  


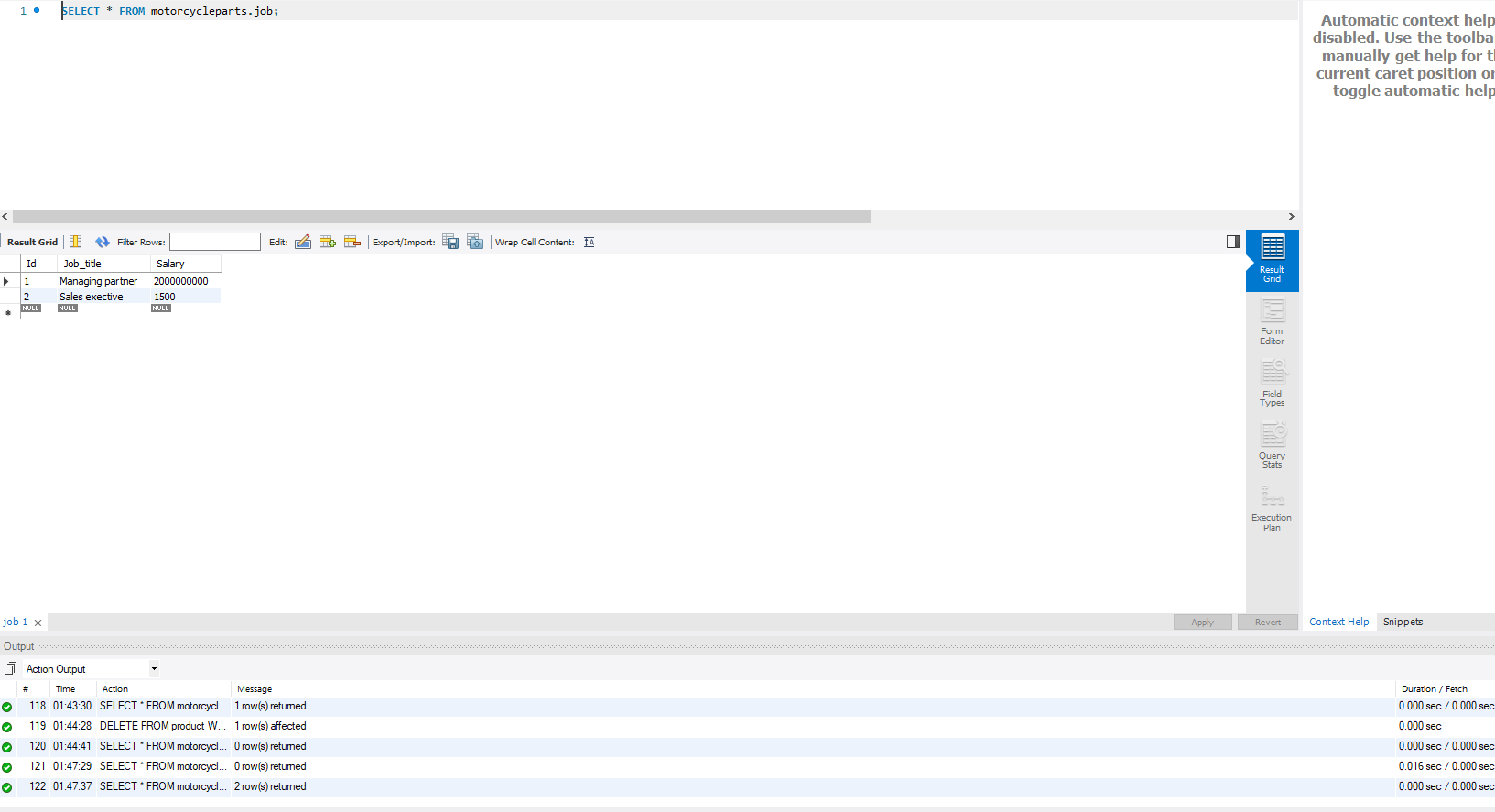
UPDATE job SET Salary = 1500

WHERE Id=2;



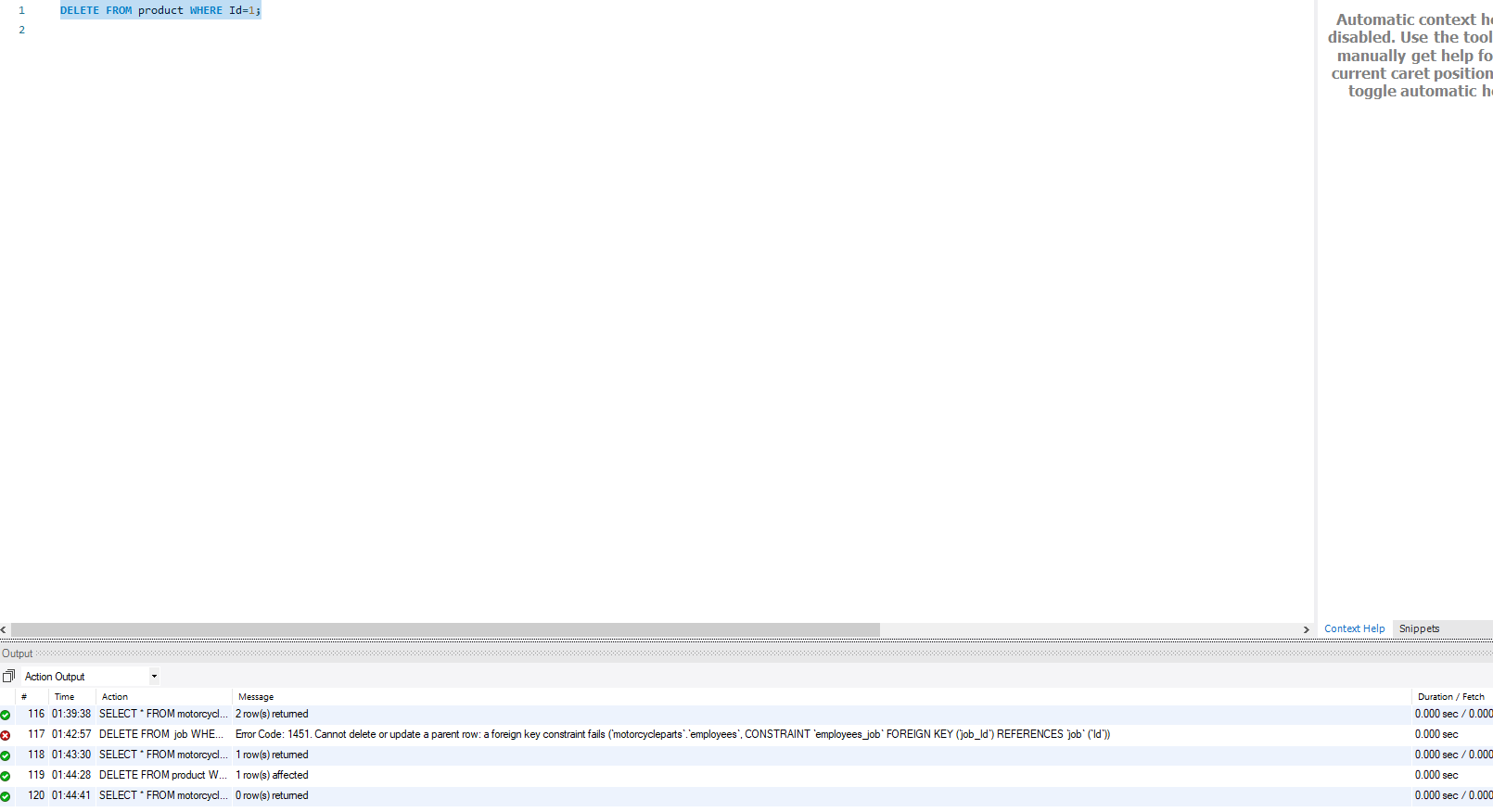
SELECT:

SELECT \* FROM motorcycleparts.job;



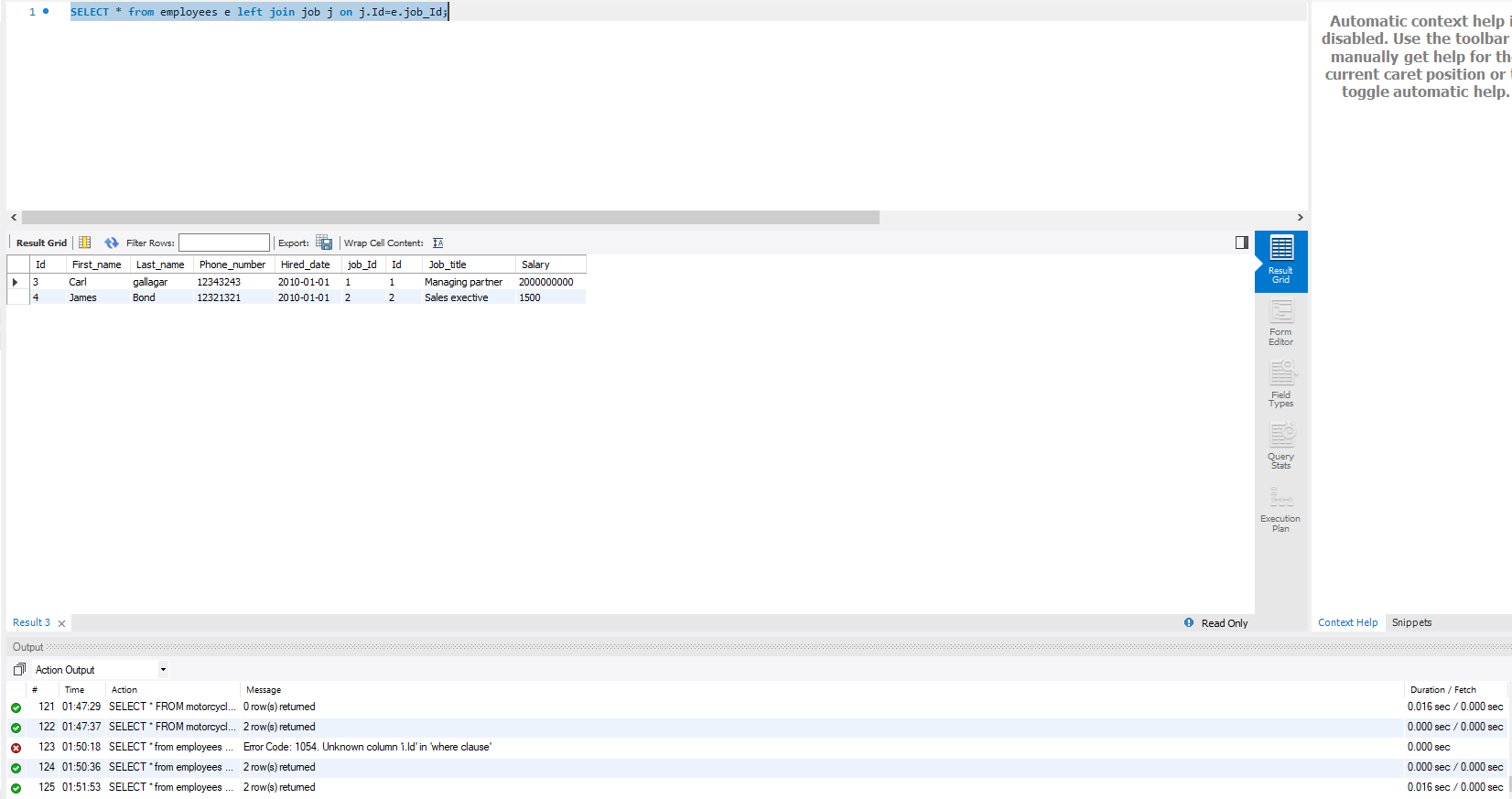
DELETE:

DELETE FROM product WHERE Id=1;

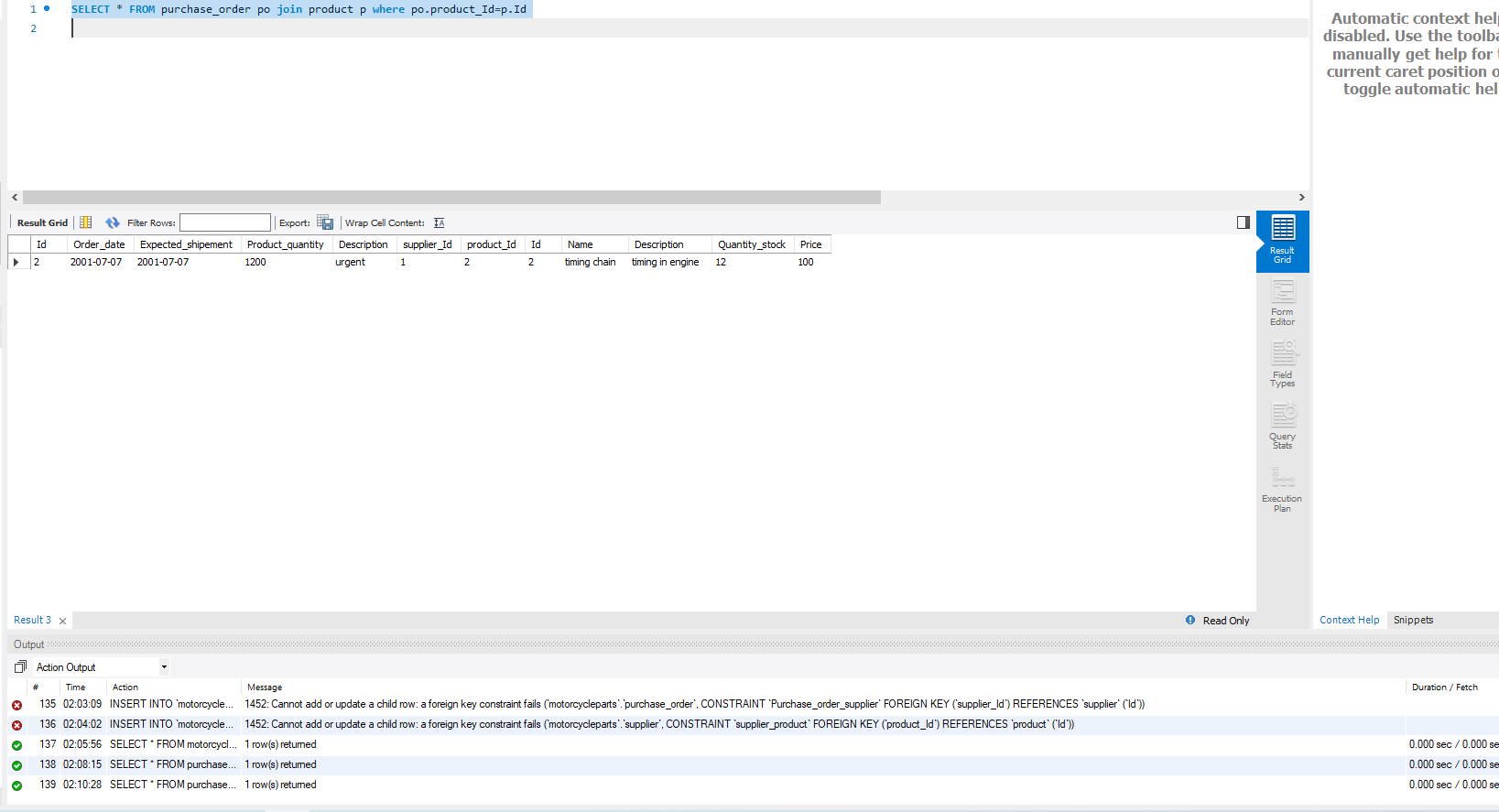


**JOIN STATEMENTS**

SELECT \* from employees e left join job j on j.Id=e.job\_Id;

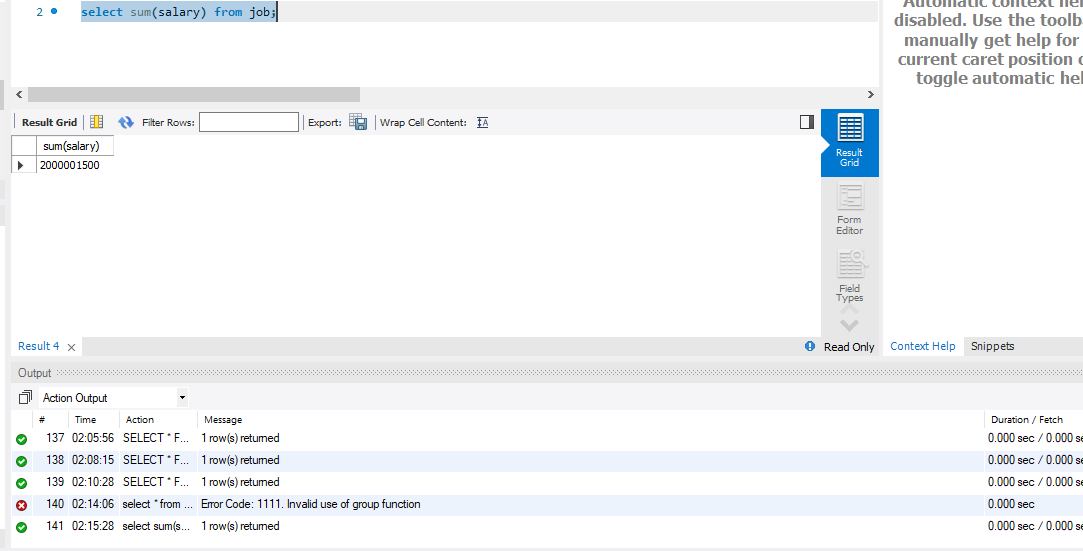


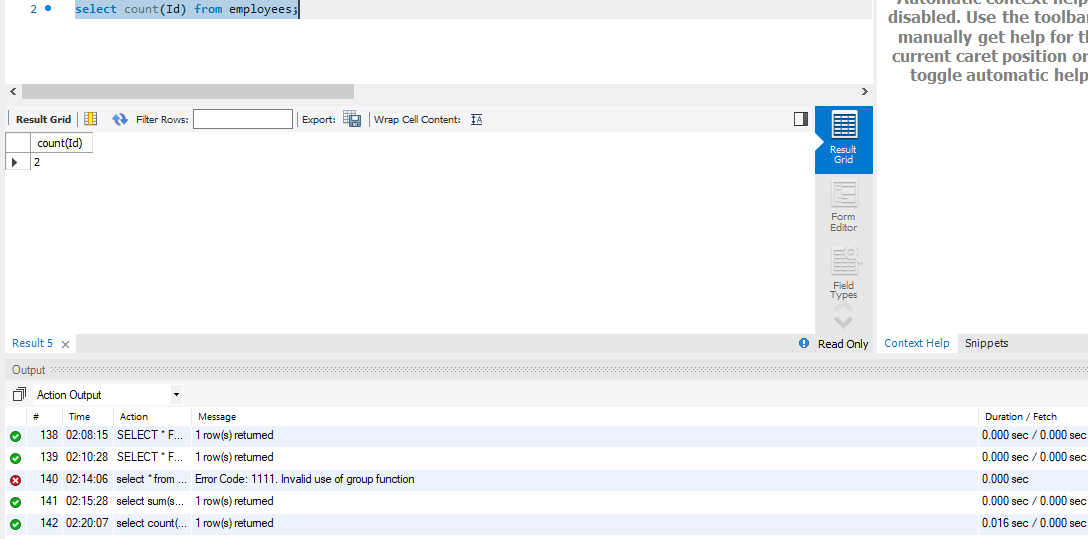
SELECT \* FROM purchase\_order po join product p where po.product\_Id=p.Id



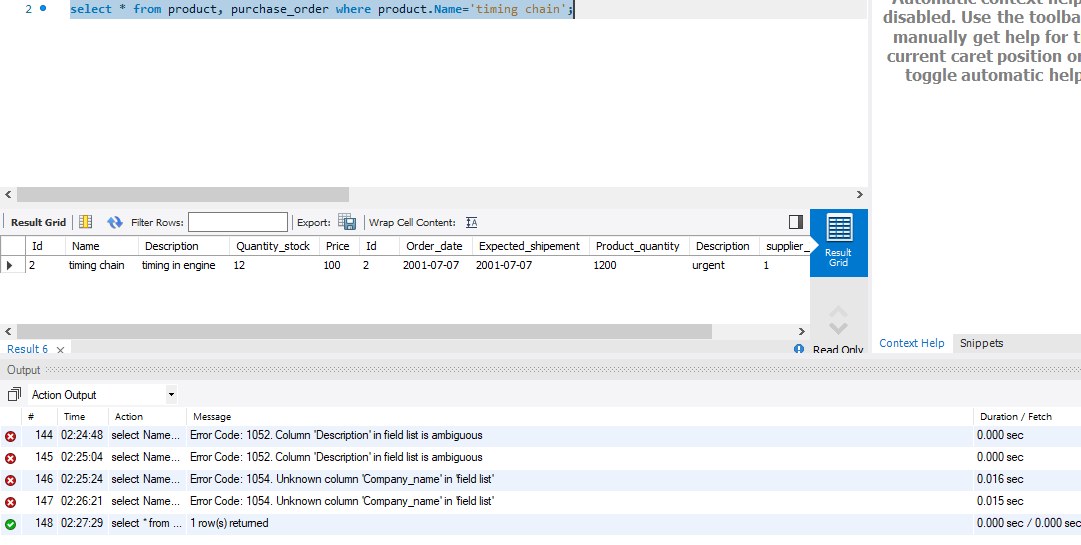
SUMMARY STATEMENTS:

select sum(salary) from job;

m job;   
select count(Id) from employees;

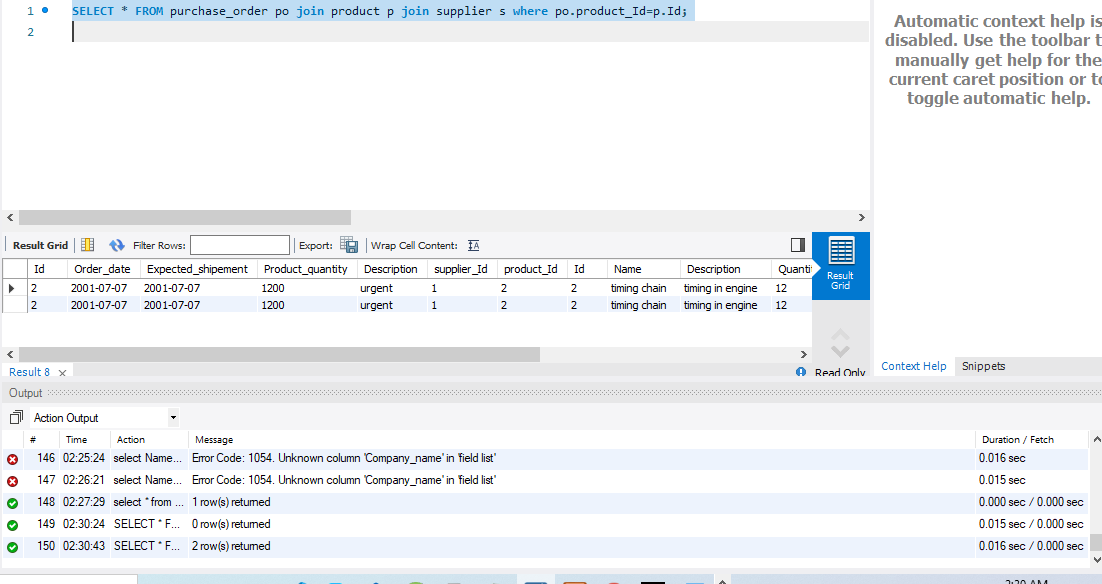


MULTIPLE TABLE QUERY:  
select \* from product, purchase\_order where product.Name='timing chain';



QUERY OF CHOICE:

SELECT \* FROM purchase\_order po join product p join supplier s where po.product\_Id=p.Id;



**Indexes**

Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries. (what tests could you run? Remember the slow query log?) . You can put all of this information in your **report.**

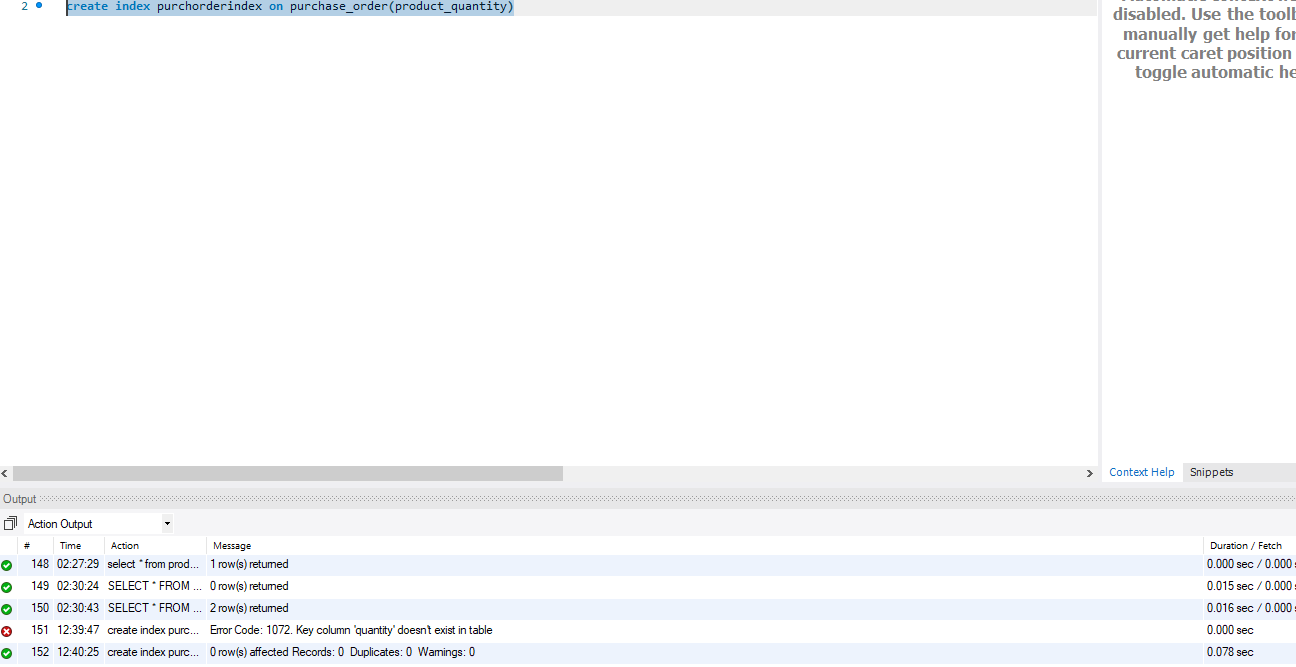
Rubric: Your work will be graded as follows:

* 6 points for clearly defining at least three indexes and explaining why you chose them.
* 3 points for showing the sql needed to generate the indexes
* 3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.

Total points possible: 12

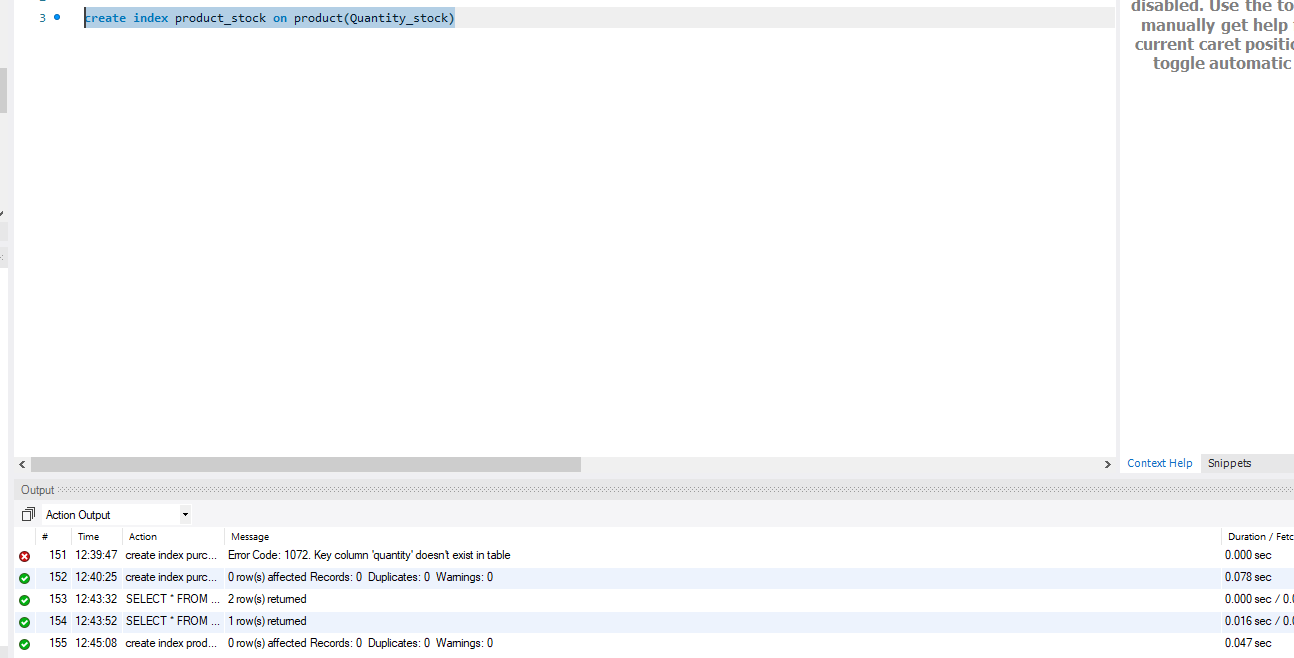
1. **create index purchorderindex on purchase\_order(product\_quantity);**

It will improve the select statement on purchase\_order table when grouped by product quantity



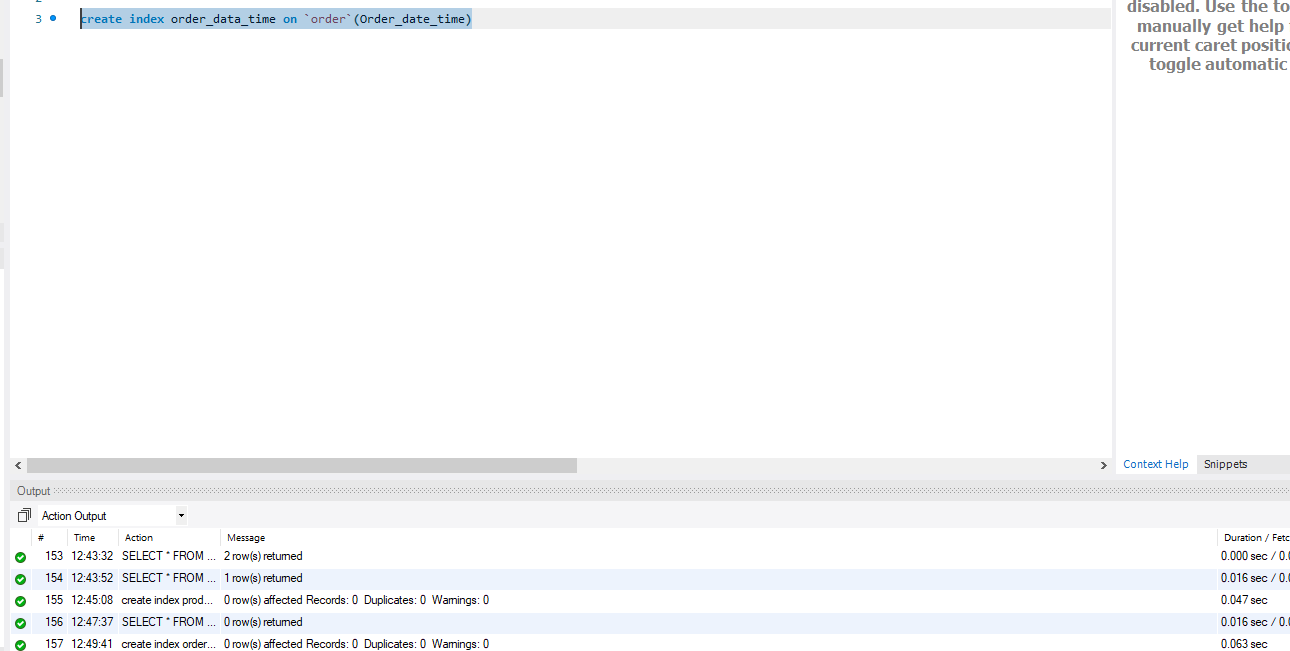
**2.create index product\_stock on product(Quantity\_stock)**

It will create index on product table on quantity stock which will make stock collection faster

****

1. **create index order\_data\_time on `order`(Order\_date\_time)**

This will improve performance on date and time of order



**Views**

Add two views to your database to provide easy access to combinations of data from multiple tables. Document the work and screenshots in your **report**.

Rubric: Your work will be graded as follows:

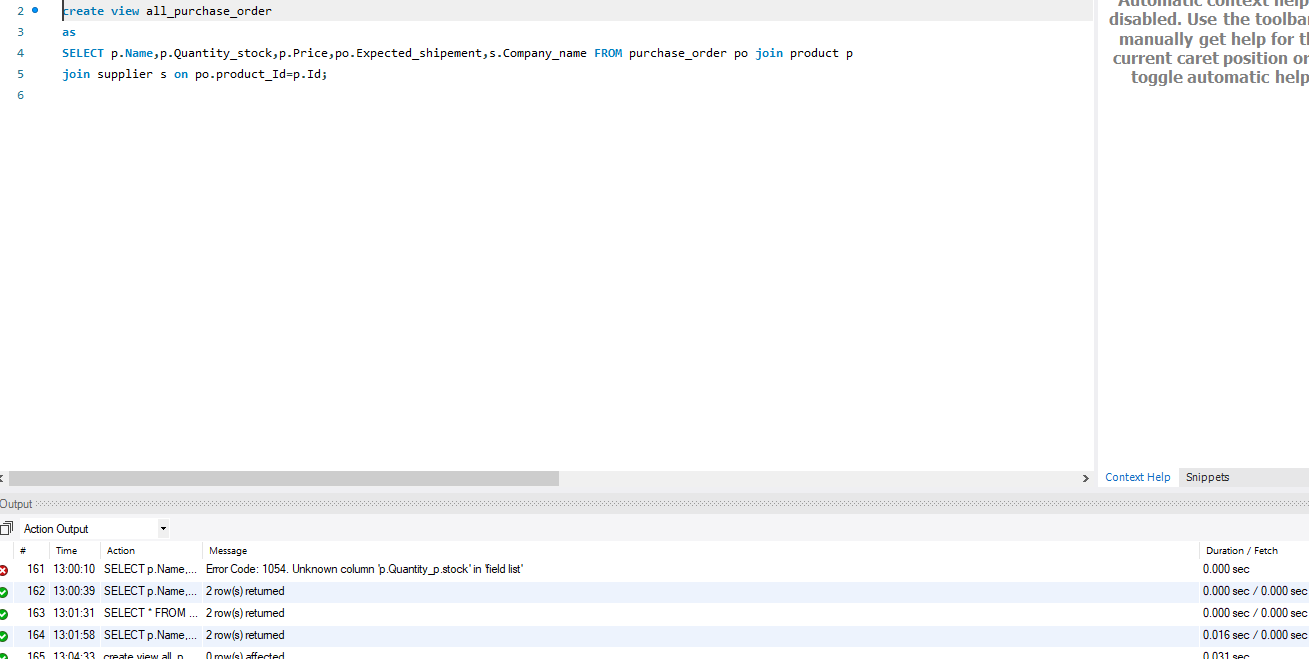
* 2 points for including the SQL for generating the two views in your Word document
* 2 points for including screenshots for the data contained in each view in your **Word document**
* 2 points for explaining why each view is a valuable addition to your database
* 2 points for explaining who might benefit most from having access to each view.

Total points possible: 8

1. Create view all\_purchase\_order as SELECT p.Name,p.Quantity\_stock,p.Price,po.Expected\_shipement,s.Company\_name FROM purchase\_order po join product p

join supplier s on po.product\_Id=p.Id;

This view will be used by admin when viewing purchase order to get a wholistic view of purchase



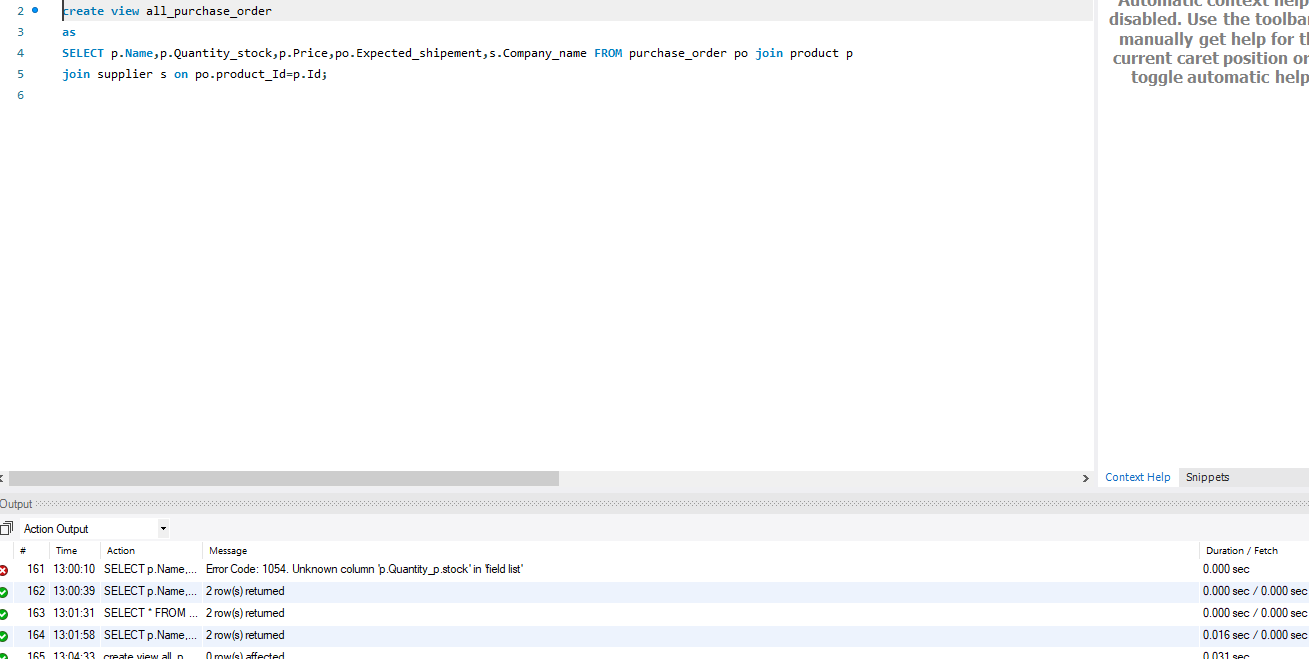
1. create view customer\_buys

as

SELECT c.First\_name,c.Phone\_number,o.Order\_quantity,o.Order\_date\_time

FROM customer c join `order` o on c.Id=o.Customer\_Id;

This view will be used in viewing customer along with their orders



**Triggers**

Add a trigger to a table so that data will be updated when a certain event occurs. Document your triggers and screenshots in your **report**.

Rubric: Your work will be graded as follows:

* 2 points for including the SQL for the trigger in your Word document
* 2 points for clearly explaining the purpose of the trigger
* 2 points for a screenshot and explanation that shows the trigger in action.

Total points possible: 6

create trigger totall

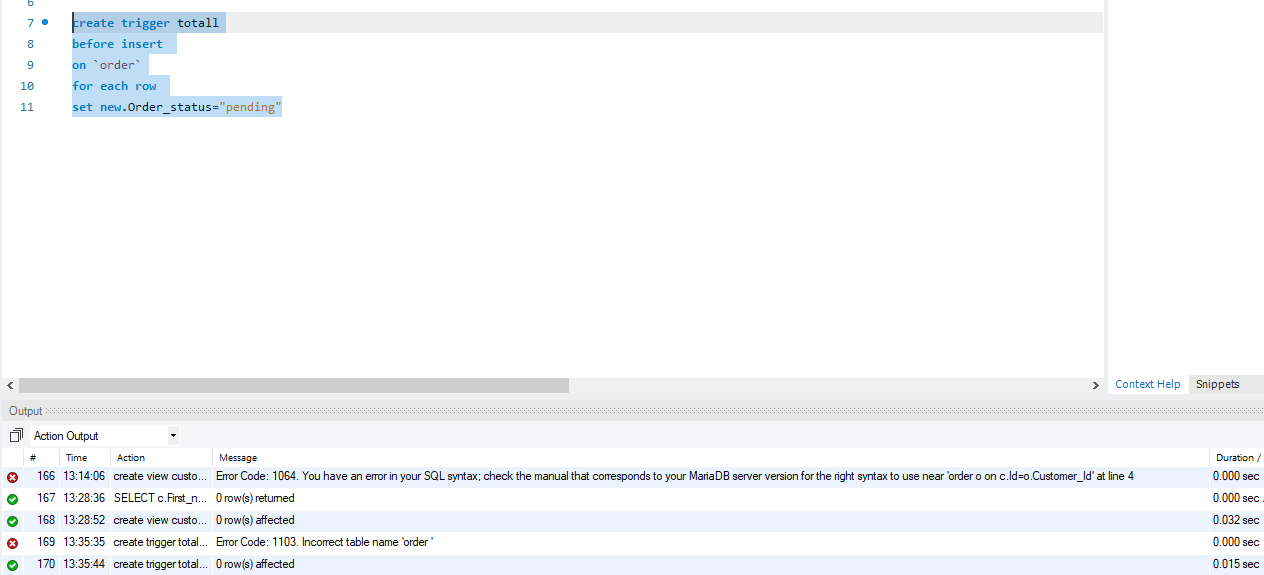
before insert

on `order`

for each row

set new.Order\_status="pending"

purpose: This trigger will turn every order entered to pending



**Transactions**

Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior? Document all of this in your **report**.

Rubric: Your work will be graded as follows:

* 3 points for clearly explaining the importance of transactions to ensuring ACID behavior
* 3 points for including a screenshot and accompanying explanation of a MySQL transaction.

Total points possible: 6

In the example below transaction ensures the atomicity, consistency, isolation, and durability. Each of these qualities contribute towards ensuring the integrity of data. Either all of the operations in a transaction are performed or none at all.

Example:

BEGIN;

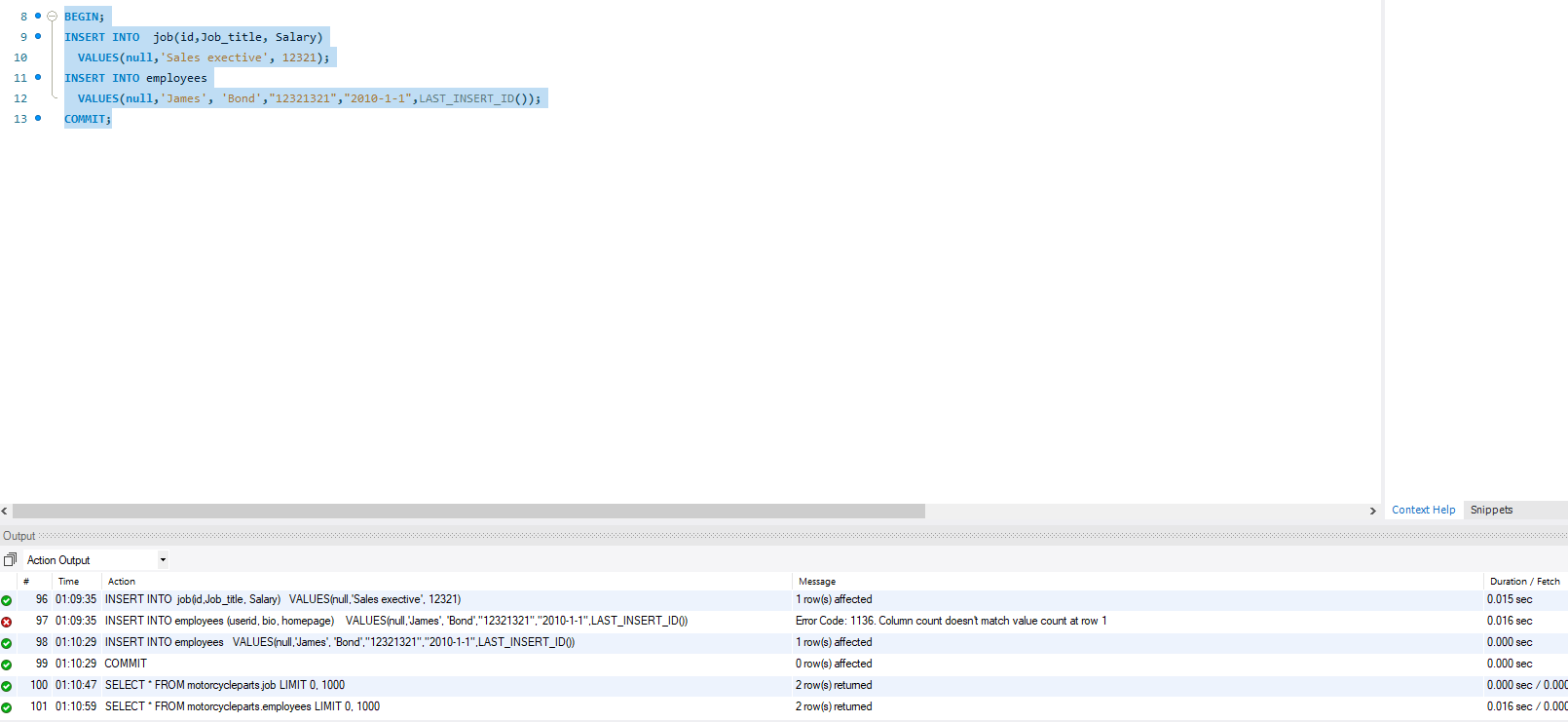
INSERT INTO job(id,Job\_title, Salary)

VALUES(null,'Sales exective', 12321);

INSERT INTO employees

VALUES(null,'James', 'Bond',"12321321","2010-1-1",LAST\_INSERT\_ID());

COMMIT;



**Security**

Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users. Document all of this in your **report.**

Rubric: Your work will be graded as follows:

* 6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.
* 4 points for writing GRANT statements that assign privileges to these different kinds of users.
* 4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.

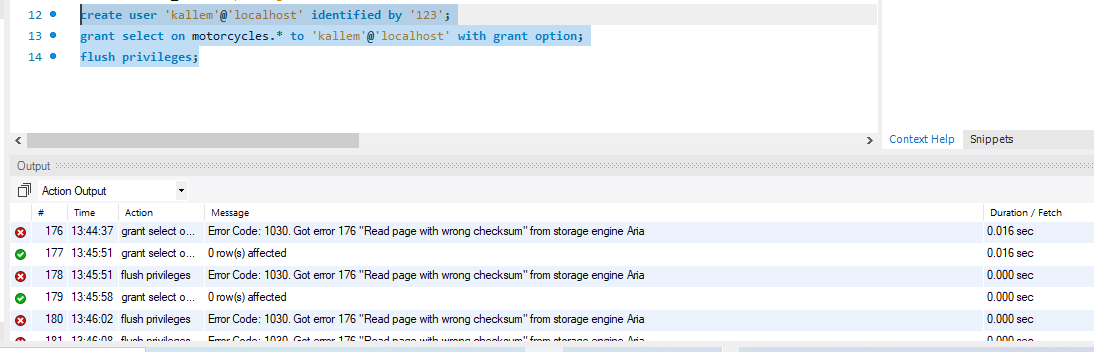
Total points possible: 14

1. Root user/admin:  
   Admin will have all the privileges including the grant command itself.
2. Employee: employee with select privilege only

create user 'kallem'@'localhost' identified by '123';

grant select on motorcycles.\* to 'kallem'@'localhost' with grant option;

flush privileges;



**Locking**

Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place. Document all of this in your **report.**

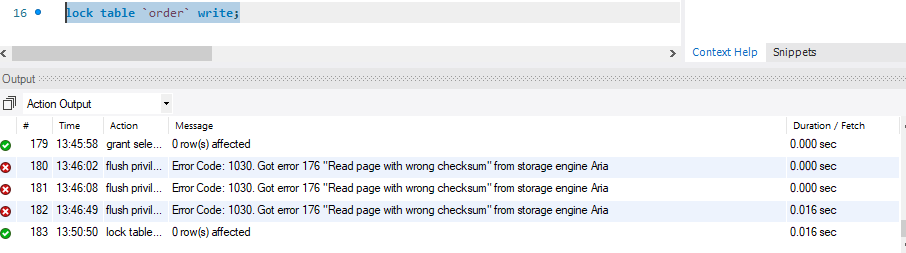
Rubric: Your work will be graded as follows:

* 3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.
* 3 points for providing a screenshot and accompanying explanation of locking tables.

Total points possible: 6

MySQL provides the facility to every client session to acquire the table by locking it to prevent the other client sessions to use that table

lock table `order` write;



**Backup**

How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored? Document all of this in your **report.**

Rubric: Your work will be graded as follows:

* 12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)
* 3 points for providing a screenshot of the command you would issue to back up the database.

Total points possible: 15

I will backup database every 6 months depending on quantity of data making it secure. I will store it on cloud server

Command to backup database

mysqldump -h localhost -u root -p “” motorcycleparts > motorcycleparts.sql

**Python Programming**

Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document **report**, and also post the program to your **GitHub** repository.  
  
The report is just a SELECT \* command that displays the contents of each table. If you have many records, you might want to look into the LIMIT option to restrict your output. Since the previous parts of this assignment specify 3 records, your output should be anywhere from 3 to 10 records per table.

Rubric: Your work will be graded as follows:

* 12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code (the .py file) must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.
* 2 points for posting the code to GitHub
* 4 points for showing a screenshot of your running the script and showing the results it produces on the screen.

Total points possible: 18

**--**

**import mysql.connector**

**from mysql.connector import errorcode**

**# a function that takes connection and query and return everything in the list**

**def select(conn,query):**

**cursor = conn.cursor()**

**cursor.execute(query)**

**results = []**

**for row in cursor.fetchall():**

**results.append(row)**

**cursor.close()**

**return results**

**# a function that takes connection and query and commits the query**

**def execute(conn,query): # update, delete, and insert**

**cursor = conn.cursor()**

**cursor.execute(query)**

**conn.commit()**

**# a function that prints the resulttothe screen**

**def show(rows):**

**for row in rows:**

**print(row)**

**# trying to establish connection to the database**

**try:**

**conn = mysql.connector.connect(**

**user="root",**

**password="",**

**host="localhost",**

**database="motorcycleparts")**

**except mysql.connector.Error as err:**

**print("Cannot connect.")**

**exit()**

**# storing query in variable**

**# function call**

**rows = select(conn,"select \* from customer")**

**# function call**

**show(rows)**

**rows = select(conn,"select \* from employees")**

**# function call**

**show(rows)**

**rows = select(conn,"select \* from job")**

**# function call**

**show(rows)**

**rows = select(conn,"select \* from product")**

**# function call**

**show(rows)**

**rows = select(conn,"select \* from purchase\_order")**

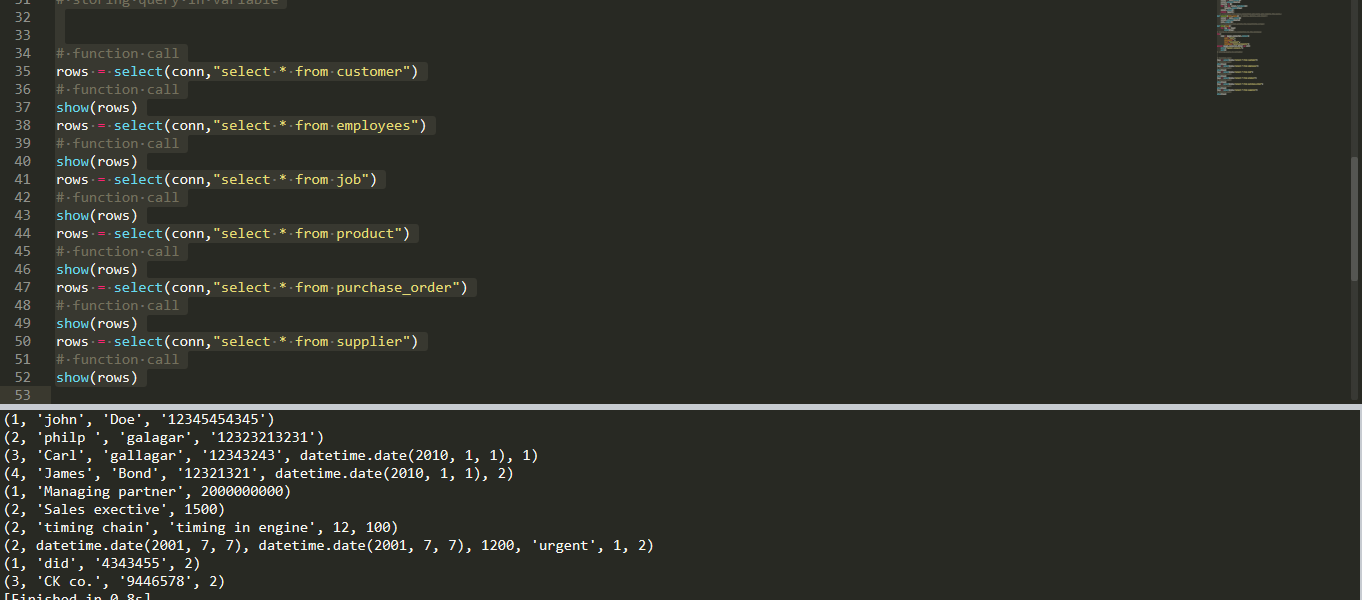
**# function call**

**show(rows)**

**rows = select(conn,"select \* from supplier")**

**# function call**

**show(rows)**

****

**BONUS POINTS:**

If you complete all of the other sections, then you may try this section for bonus points. Feel free to do the PHP or the “Suggested Future Work”, or both!  
 **PHP Programming**

Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.

Rubric: Your work will be graded as follows:

* 4 points for writing an HTML form the user will use to enter search criteria
* 8 points for a PHP script that uses the search criteria and returns results
* 4 points for an HTML page that shows the results
* 4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.
* 4 points for providing screen shots of your PHP website in action.
* 2 points for posting your code to GitHub

Total points possible: 26

**Suggested Future Work**

Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.  
  
This requires researching what NoSQL is, what makes it different from MySQL, explaining how the 2 systems are different, and providing APA citations of your sources. You don’t need to create a separate bibliography or references section, but do provide the formatted source at the bottom of the document.

Tip: Purdue has an automatic citation generator page that is incredibly useful for this function.

Rubric: Your work will be graded as follows:

* 3 points for clearly describing the limitations of your database design
* 3 points for explaining how you would address these shortcomings
* 3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.
* 3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.
* 2 points for the source references.

Total points possible: 14

Grand Total:

245 + 40 bonus