Lab Manual on Stored Routines and Views (November 2024)

Allan O. Omondi
School of Computing and Engineering Sciences,
Strathmore University, Nairobi, Kenya
aomondi@strathmore.edu

Learning Outcomes

By the end of this lab, you will be able to:

1. **Differentiate** between a traditional programming approach and a data science approach to problem-solving
2. **Differentiate** between data science and Business Intelligence
3. **Implement** a data-processing-related algorithm using SQL
4. **Apply** an SQL LOOP to iterate through sections of code
5. **Apply** an SQL CURSOR to iterate through each row of a result set
6. **Create, view, and delete** a procedure
7. **Create, view, and delete** a function
8. **Create** and **delete** a view
9. **Invoke** a procedure
10. **Invoke** a function
11. **Use** an SQL view

Original GitHub Repository

https://github.com/course-files/BBT3104-Lab5of6-StoredRoutinesandViews

Software

1. VS Code: link
2. Git and Git Bash: link
3. GitHub Desktop: link
4. WSL (for Windows OS): link
5. Docker: link
6. DBeaver database tool: link
7. VS Code extensions: Docker, WSL (for Windows OS), YAML, Code Spell Checker

Approximate Time Required

1 hour

Prerequisites

1. You need to have completed Lab 3 on Scheduled Events

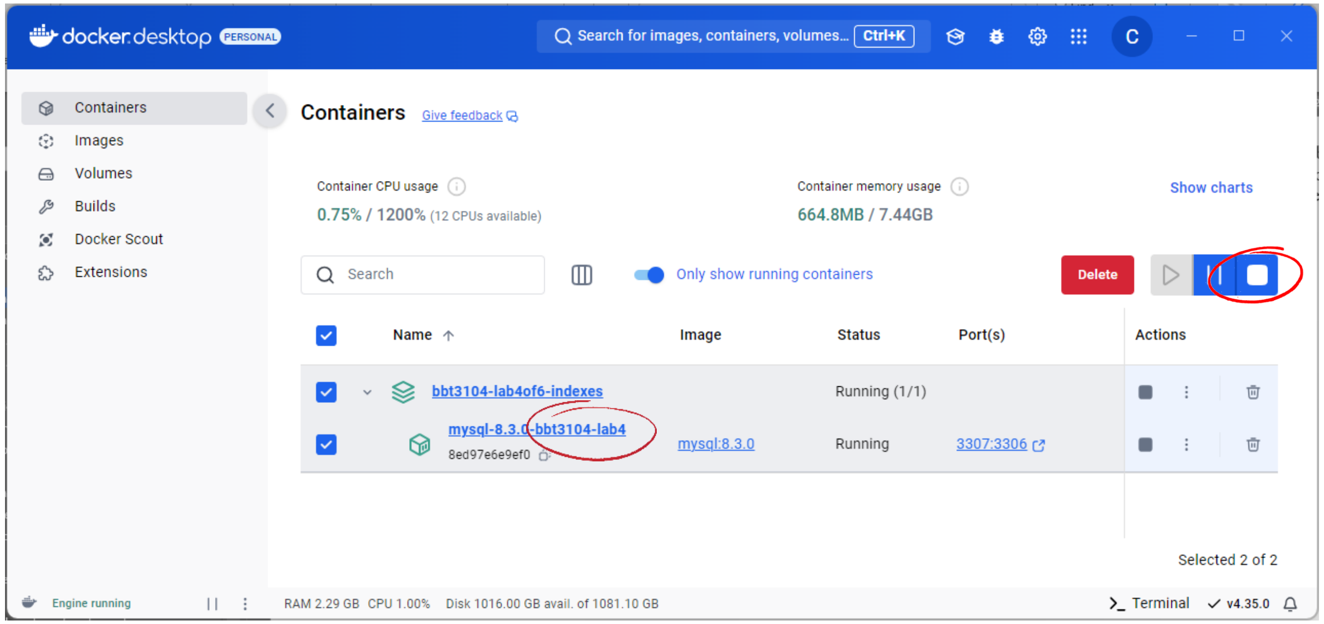
Recommended citation:
Omondi, A. O. (2024). *BBT3104 – Advanced Database Systems – Lab Manual on Stored Routines and Views [PDF Lab Manual]*. Retrieved from the Strathmore University eLearning website https://elearning.strathmore.edu/

CC BY-NC-SA 4.0

Practical Steps

STEP 1. Stop the MySQL Docker container used in the previous lab

Stop the `mysql-8.3.0-bbt3104-lab4` container we used in Lab 2. This is to avoid conflicts when creating another container in lab 4 that is using the same port in the host, i.e., port 3307. Remember that we can specify a port number in the host so that any data transmitted through this port is relayed to the container. For example, if data is sent to the host through port 3307, this data is relayed to the container through the container's port 3306.



Containers

Container CPU usage 0.75% / 1200% (12 CPUs available)

Container memory usage 664.8MB / 7.44GB

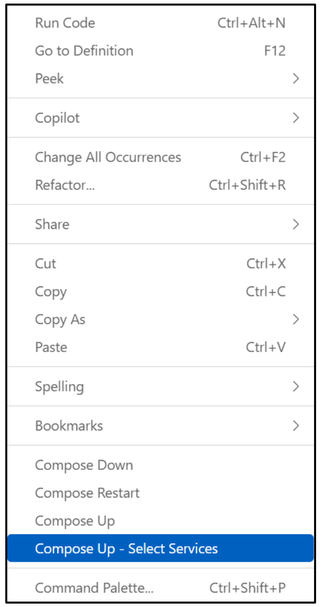
Name	Image	Status	Port(s)	Actions
bbt3104-lab4of6-indexes		Running (1/1)		
mysql-8.3.0-bbt3104-lab4 8ed97e6e9ef0	mysql:8.3.0	Running	3307:3306	

Selected 2 of 2

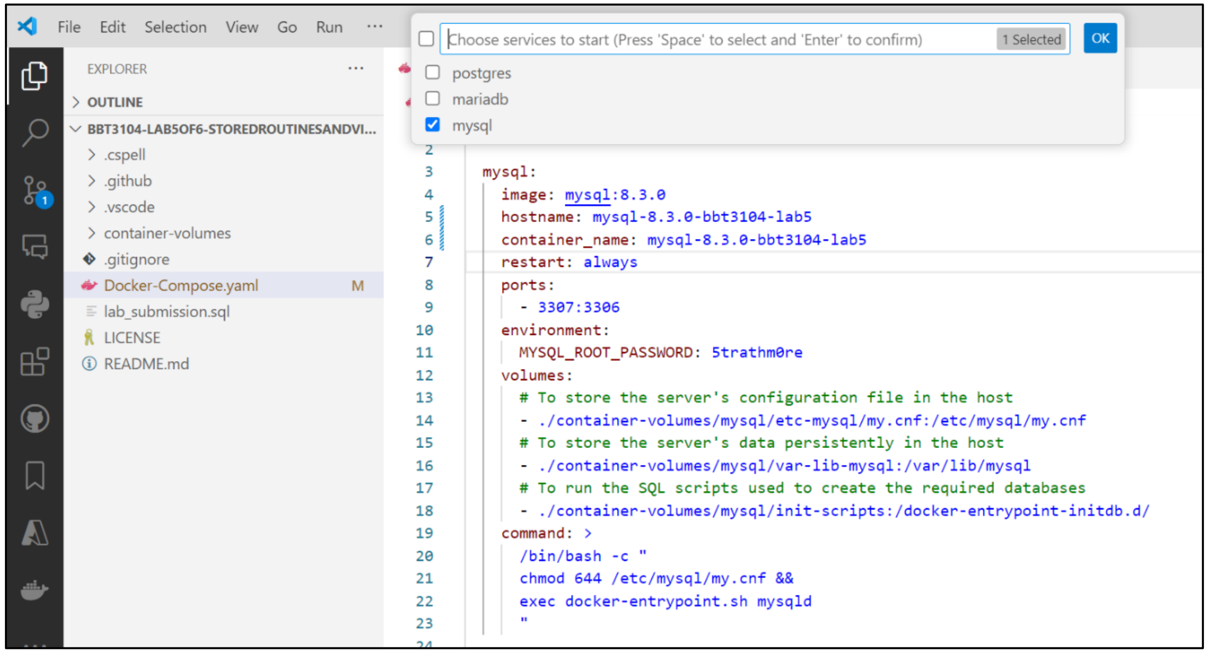
RAM 2.29 GB CPU 1.00% Disk 1016.00 GB avail. of 1081.10 GB

STEP 2. Start the MySQL Docker container for Lab 4

The `Docker-Compose.yaml` code is available here: https://github.com/course-files/BBT3104-Lab5of6-StoredRoutinesandViews/blob/main/Docker-Compose.yaml

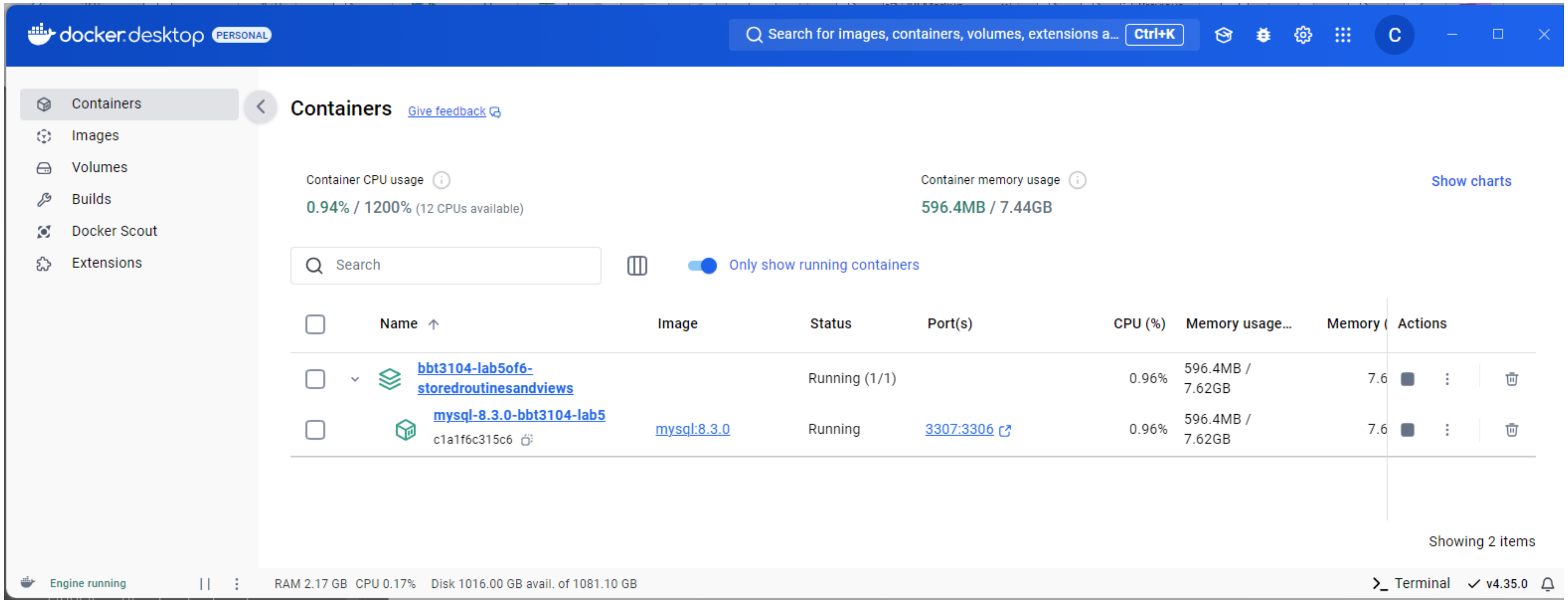


Right-Click anywhere inside the `Docker-Compose.yaml` file in VS Code and select "**Compose Up – Selected Services**"



```
mysql:
  image: mysql:8.3.0
  hostname: mysql-8.3.0-bbt3104-lab5
  container_name: mysql-8.3.0-bbt3104-lab5
  restart: always
  ports:
    - 3307:3306
  environment:
    MYSQL_ROOT_PASSWORD: 5trathm0re
  volumes:
    # To store the server's configuration file in the host
    - ./container-volumes/mysql/etc-mysql/my.cnf:/etc/mysql/my.cnf
    # To store the server's data persistently in the host
    - ./container-volumes/mysql/var-lib-mysql:/var/lib/mysql
    # To run the SQL scripts used to create the required databases
    - ./container-volumes/mysql/init-scripts:/docker-entrypoint-initdb.d/
  command: >
    /bin/bash -c "
    chmod 644 /etc/mysql/my.cnf &&
    exec docker-entrypoint.sh mysqld
    "
```

Ensure that Docker is running in the background then select "mysql" and click "OK" to create the MySQL Docker container.



Containers Give feedback

Container CPU usage
0.94% / 1200% (12 CPUs available)

Container memory usage
596.4MB / 7.44GB

Show charts

Only show running containers

Name	Image	Status	Port(s)	CPU (%)	Memory usage...	Memory (
bbt3104-lab5of6-storedroutinesandviews		Running (1/1)		0.96%	596.4MB / 7.62GB	7.6
mysql-8.3.0-bbt3104-lab5 c1a1f6c315c6	mysql:8.3.0	Running	3307:3306	0.96%	596.4MB / 7.62GB	7.6

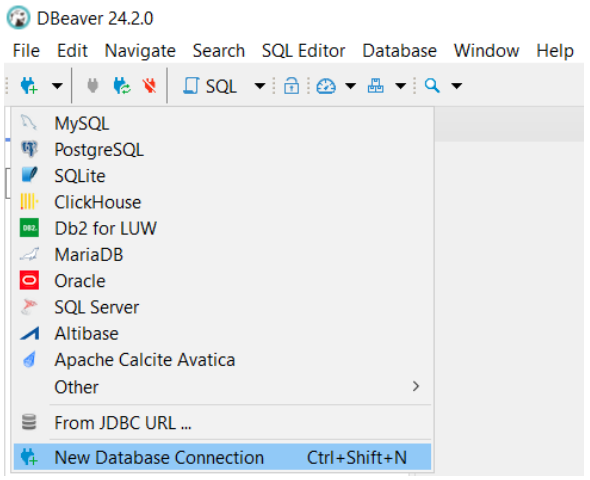
Showing 2 items

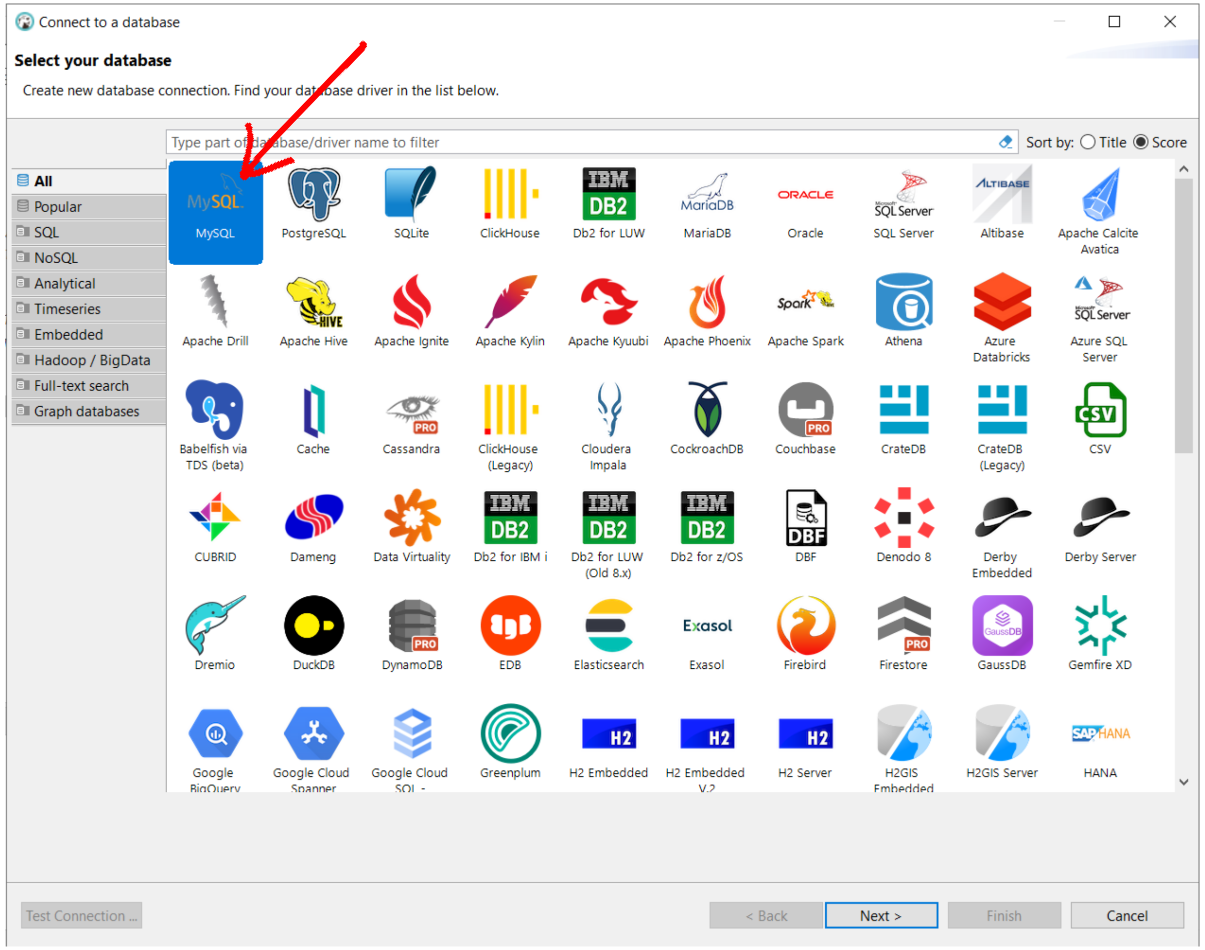
Engine running RAM 2.17 GB CPU 0.17% Disk 1016.00 GB avail. of 1081.10 GB v4.35.0

The assigned name of the MySQL Docker container is “`mysql-8.3.0-bbt3104-lab5`” and it should appear as “**Running**” in the list of containers in Docker Desktop as shown here. Note that it is listed under the cluster called “`bbt3104-lab5of6-storedroutinesandviews`”. The name of the cluster is assigned based on the folder name that is storing the `Docker-Compose.yaml` file.

STEP 3. Create a connection to the MySQL container using DBeaver database tool

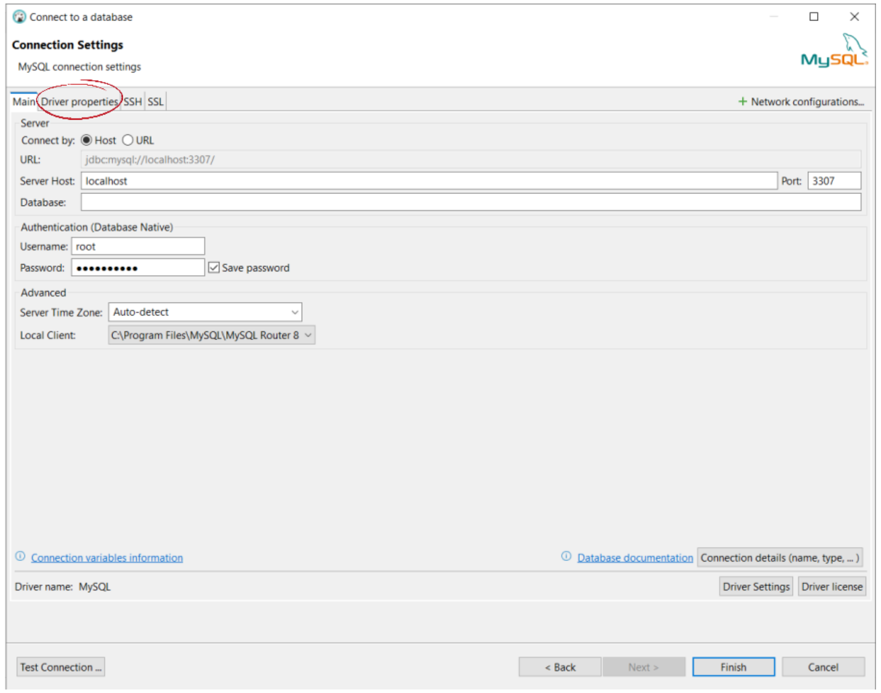
Alternatively, you can use the existing connection you created in Lab 1.





Open DBeaver and create a new connection.

Select the MySQL standard driver and click “Next”.


Connect to a database

Connection Settings

MySQL connection settings

Main | Driver properties | SSH | SSL   + Network configurations...

Server

Connect by: (●) Host ( ) URL

URL: jdbc:mysql://localhost:3307/

Server Host: localhost   Port: 3307

Database:

Authentication (Database Native)

Username: root

Password: ●●●●●●●●●●   ☑ Save password

Advanced

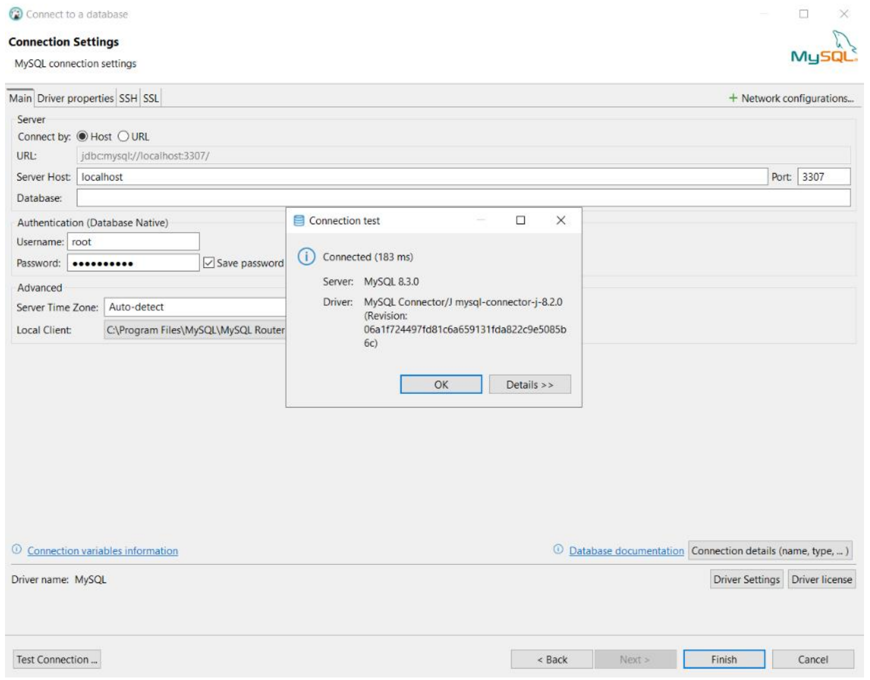
Server Time Zone: Auto-detect

Local Client: C:\Program Files\MySQL\MySQL Router 8

Connection variables information   Database documentation   Connection details (name, type, ...)

Driver name: MySQL   Driver Settings   Driver license

Test Connection ...   < Back   Next >   Finish   Cancel


Connection test

Connected (183 ms)

Server: MySQL 8.3.0

Driver: MySQL Connector/J mysql-connector-j-8.2.0 (Revision: 06a1f724497fd81c6a659131fda822c9e5085b6c)

OK   Details >>

Specify the server's IP address (localhost), port number (3307), username (root or student), and password (5trathm0re).

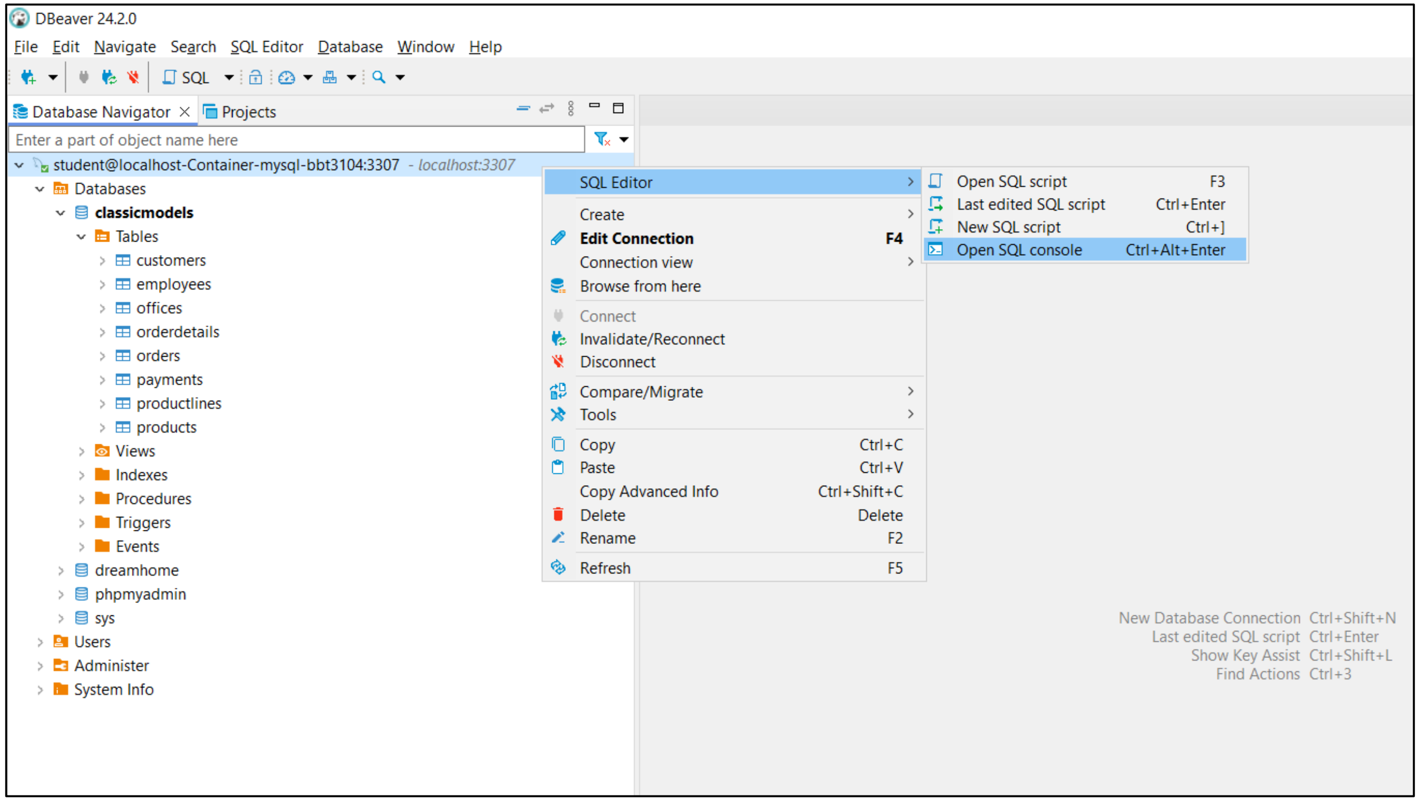
Test the connection settings and click "Finish" if the test is successful. You do not need to change any other settings.

Set the value of "`allowPublicKeyRetrieval`" property under the "Driver Properties" tab to TRUE. Create the property if it does not exist.

Remember to open the ports in the firewall if you are using Linux, i.e., execute the following in the VS Code Git Bash terminal:

```
sudo ufw allow 3307/tcp
```

Open an SQL console using the connection:



STEP 4. Create a connection to the MySQL container using the Command Line Interface

Execute the following in VS Code Git Bash terminal to connect to the MySQL container using its command line interface:

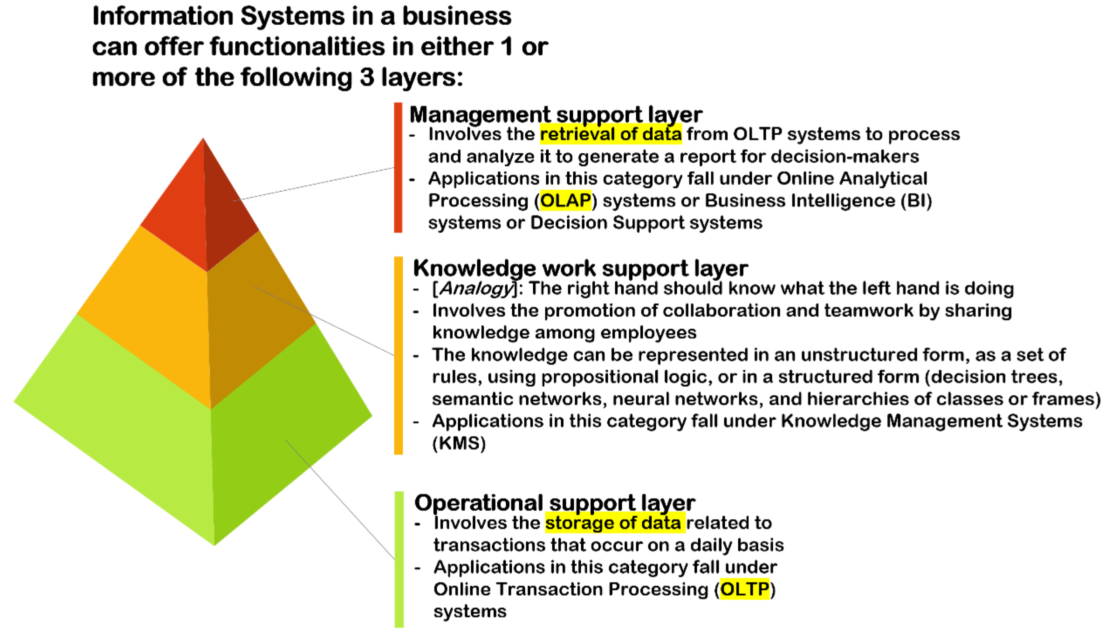
```
docker exec -it mysql-8.3.0-bbt3104-lab5 mysql -u student -p
```

Narrative

The "classicmodels" sample database contains data of a retail business. The retail business, in this case, is engaged in the sale of **models of** classic cars, motorcycles, planes, ships, trains, trucks and buses, and vintage cars. It contains typical business data such as customers, products, product lines (categories of products), orders, order line items (details of an order), payments received, employees, and branch details. The database schema is presented here: https://github.com/course-files/BBT3104-Lab4of6-Indexes/blob/main/container-volumes/mysql/init-scripts/classicmodels.png

The Data, Information, Knowledge, Action, Result (DIKAR) Model

The storage of data and the retrieval of data are key functionalities that are required in the operational support and management support layers of a business information system respectively.



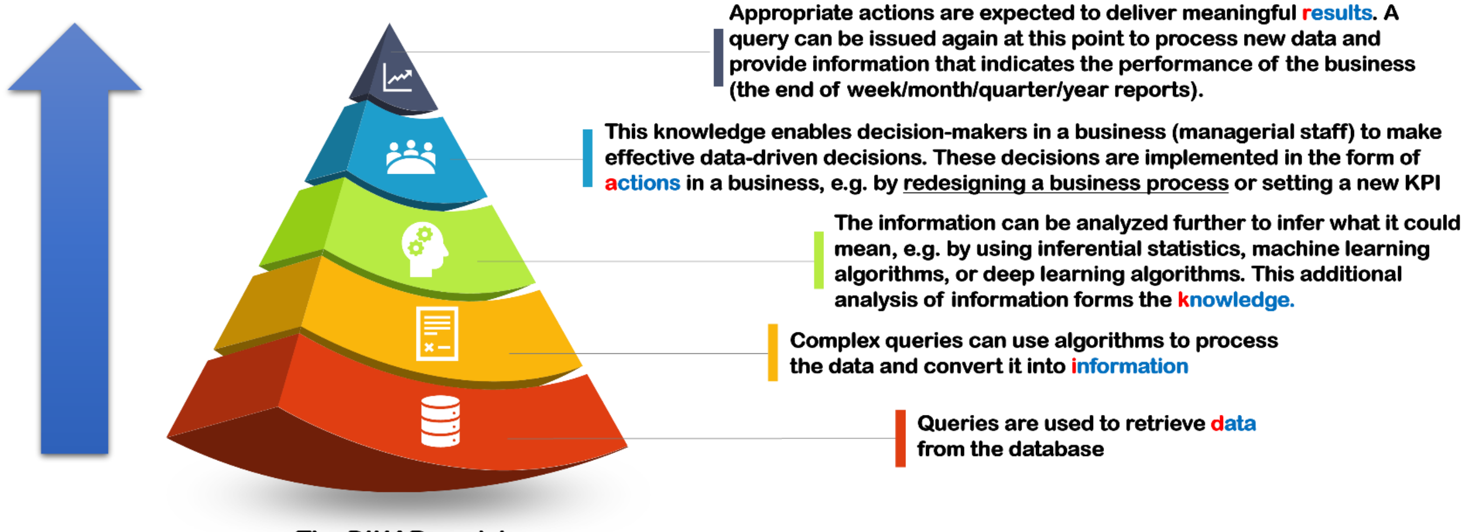
Information Systems in a business can offer functionalities in either 1 or more of the following 3 layers:

Management support layer
- Involves the retrieval of data from OLTP systems to process and analyze it to generate a report for decision-makers
- Applications in this category fall under Online Analytical Processing (OLAP) systems or Business Intelligence (BI) systems or Decision Support systems

Knowledge work support layer
- [*Analogy*]: The right hand should know what the left hand is doing
- Involves the promotion of collaboration and teamwork by sharing knowledge among employees
- The knowledge can be represented in an unstructured form, as a set of rules, using propositional logic, or in a structured form (decision trees, semantic networks, neural networks, and hierarchies of classes or frames)
- Applications in this category fall under Knowledge Management Systems (KMS)

Operational support layer
- Involves the storage of data related to transactions that occur on a daily basis
- Applications in this category fall under Online Transaction Processing (OLTP) systems

A DBA can create a complex query to retrieve data required to generate a report. This report may be required on a **routine** basis, e.g., at the end of every week or at the end of every quarter in the financial year. Expect to encounter the following terms in your career: “End of Day Report”, “End of Week Report”, “End of Month Report”, “End of Quarter Report”, and “End of Year Report”. These reports are designed to retrieve **data** from the database, which is then processed to form **information**, then analysed further (possibly using inferential statistics or machine learning algorithms or deep learning algorithms) to form **knowledge**. The knowledge enables decision-makers in the business (e.g., managerial staff) to make effective data-driven decisions which are implemented in the form of **actions** in a business. Appropriate actions are expected to deliver meaningful **results**. This is known as the DIKAR model in the fields of business strategy and information management.



Appropriate actions are expected to deliver meaningful results. A query can be issued again at this point to process new data and provide information that indicates the performance of the business (the end of week/month/quarter/year reports).
This knowledge enables decision-makers in a business (managerial staff) to make effective data-driven decisions. These decisions are implemented in the form of actions in a business, e.g. by redesigning a business process or setting a new KPI
The information can be analyzed further to infer what it could mean, e.g. by using inferential statistics, machine learning algorithms, or deep learning algorithms. This additional analysis of information forms the knowledge.
Complex queries can use algorithms to process the data and convert it into information
Queries are used to retrieve data from the database

The DIKAR model

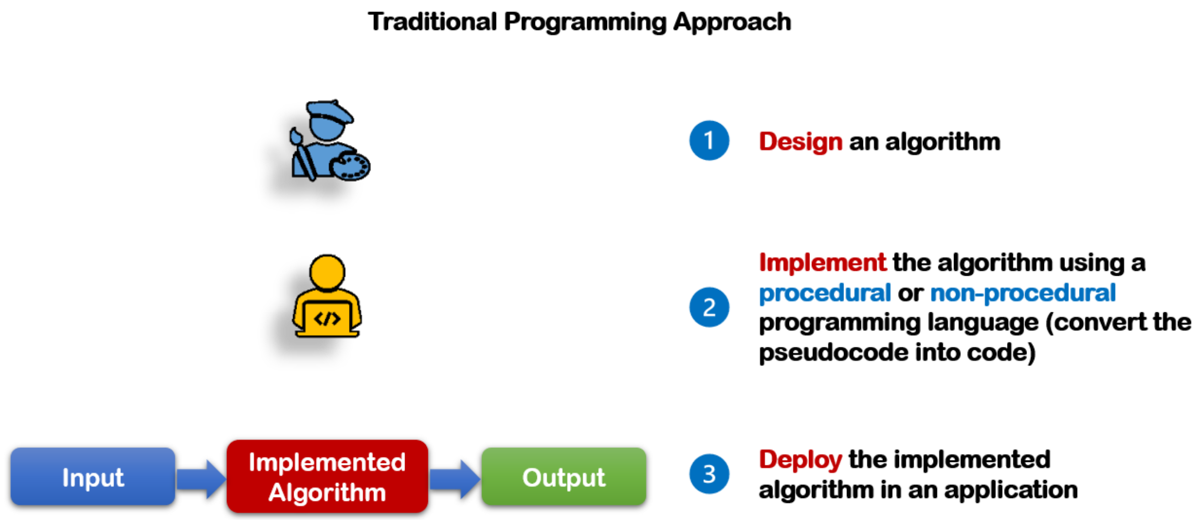
MySQL DBMS allows a DBA to store complex queries inside the database. Members of the database team can then edit or execute the complex query that is stored in a central location in the database whenever the data needs to be retrieved using the query. The database object that is used to store the complex query is called a **stored routine**. A stored routine can in turn be in the form of either a **procedure** or a **function.**

Traditional Programming Approach

Traditional programming can be grouped into either procedural approaches or non-procedural approaches. It is nowadays common to find traditional programming approaches being supplemented (not substituted) by data science which applies Machine Learning (ML). For example, a business can use a traditional programming approach to develop an e-Commerce web-based application (operational support layer) and supplement this with data science (management support layer). The data science or Business Intelligence component can in turn be used to design an association rule model that can inform the business that if a customer sees product X near product Y on the e-Commerce application, then there is a high probability that they will buy both product X and product Y. This essentially associates product X to product Y.

Data science can, therefore, be used when a traditional programming approach is inadequate to fully implement a desired task. **Given the supplementary role that data science plays, then it is still important to learn about the traditional programming approach which relies on a relational database to store structured data.**

Traditional programming involves a sequence of instructions and control flow functions (if-else, do-while, case, etc.) that determine which instruction is executed when. In some informal contexts, this setup is referred to as an "if-this-then-that" (ifttt) setup. The following diagram shows the key steps involved.


Traditional Programming Approach
1 Design an algorithm
2 Implement the algorithm using a procedural or non-procedural programming language (convert the pseudocode into code)
3 Deploy the implemented algorithm in an application
Input
Implemented Algorithm
Output

Data Science Approach
Data science involves the execution of tasks that can analyse information further to convert it into knowledge. This knowledge can be disseminated at the knowledge work support layer and used to make decisions at the management support layer of business information systems. Examples of some of the analytical tasks that are relevant to a business include:

(i) **Clustering**: Identification of groups, e.g., market segmentation. An algorithm that can be used is the k-Means clustering algorithm. For example, a business can segment the East African market into smaller segments that enable it to customize their adverts and products to suit the needs of that smaller market segment. For example, BlueBand margarine, produced by Unilever, can be customized to have a higher melting point for customers in Mtito Andei town (Makueni County) and a lower melting point for customers in Limuru town (Kiambu County). These groups can be identified by analyzing historical data that shows a common trend/desire amongst a group (cluster) of clients. This is not as straight forward as analyzing the temperature of the region.

(ii) **Classification**: Grouping items into discrete classes and predicting which class an item belongs to. For example, understanding customers can enable a business to conduct direct marketing to a group of clients that have been classified as being in the group that has a high probability of purchasing the product being marketed to them. Note that classification is **predictive** unlike clustering which is **descriptive**. Algorithms that can be used include the Naïve Bayes, adaptive Bayes network, and the Support Vector Machine algorithm.

(iii) **Association**: Looking for relationships between variables, analysing "market baskets" or customer buying patterns, and identifying items that are likely to be purchased together, for example tomatoes and onions. The business can decide to place these products side by side to encourage the client to buy both. Algorithms that can be used to identify associations include A Priori, Eclat, and FP growth

Clustering, classification, and association require the **consideration of multiple variables**. For example, it is mandatory for multi-national companies to consider details like yesterday's exchange rate; external and internal economic changes in the country that issues the currency, among other variables, when predicting the foreign exchange rate (FOREX). The number of variables to consider can be in the **hundreds** and this can be overwhelming for a traditional programming approach that relies on structured data in normalized relations inside a relational database (a database that applies a relational data model). An if-this-then-that kind of setup, therefore, may not be able to consider all the hundreds of variables accurately. The data science approach offers a different perspective that does not rely on structured data in a relational database and on designing a customized algorithm.

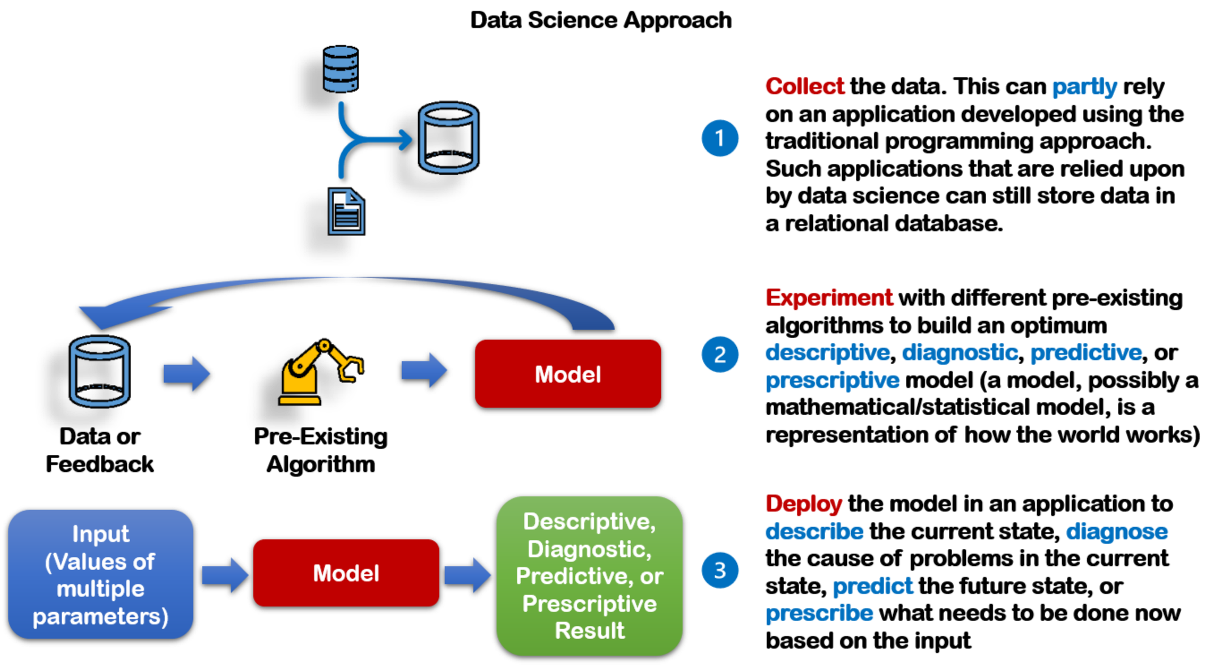
The following table shows other data models that can be used in a database apart from the relational data model.

Data Models

Data Model	Comment
Relational	Used in relational databases e.g. MySQL, Oracle, and Microsoft SQL Server. It is used to store structured data used by applications developed based on the traditional programming approach.
Key-Value	They are used to store unstructured and semi-structured data (data that is not well organized into numerous, distinct, and normalized relations that have rows and columns to organize the data). It is common to find these data models in
Graph	

Document	NoSQL databases used in the analysis of big data and in real-time applications (which started off as web 2.0). Examples of NoSQL databases include: Couchbase Server, CouchDB, Neo4j, MongoDB, Apache Cassandra, and many more listed here: https://dbdb.io/
Column-Family (or wide column store)	
Graph	
Array/Matrix	Provides input to machine learning and deep learning workloads that need to read and compare large amounts of data with speed
Vector	
Hierarchical	Considered obsolete and are currently rare to find in a production server
Network	
Multi-Value	

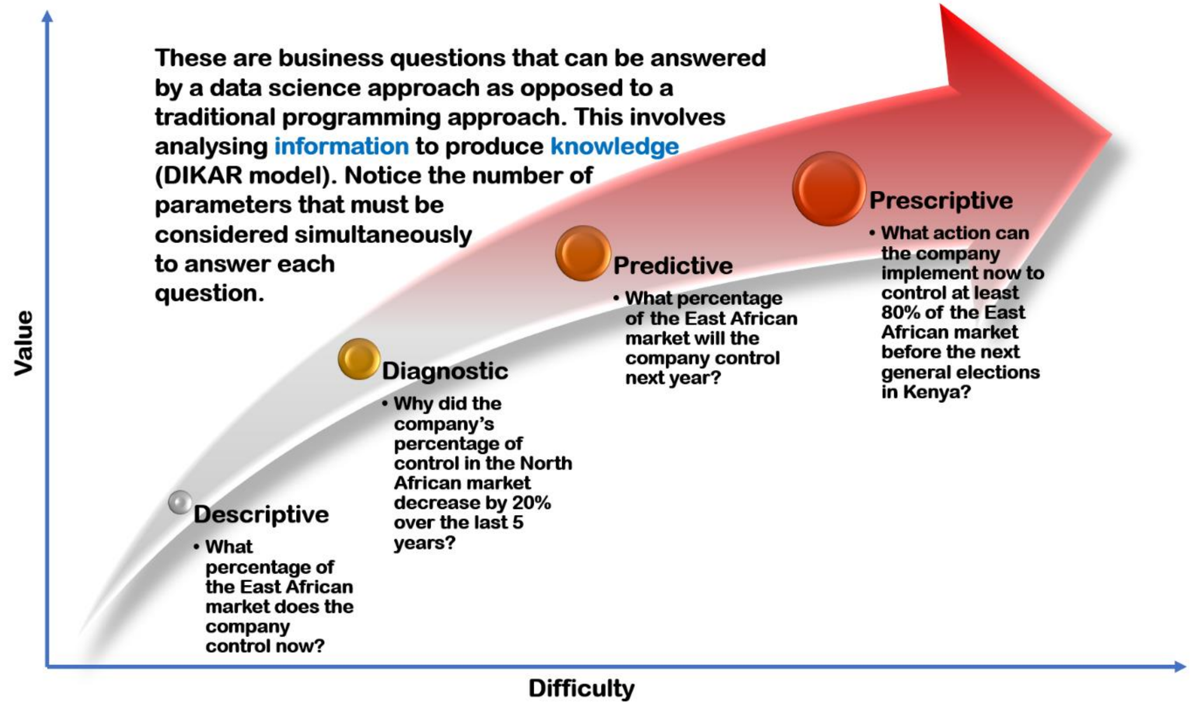
The following diagram summarizes the data science approach.


Data Science Approach
Data or Feedback
Pre-Existing Algorithm
Model
Input (Values of multiple parameters)
Model
Descriptive, Diagnostic, Predictive, or Prescriptive Result
1
2
3

1. **Collect the data. This can partly rely on an application developed using the traditional programming approach. Such applications that are relied upon by data science can still store data in a relational database.**
2. **Experiment with different pre-existing algorithms to build an optimum descriptive, diagnostic, predictive, or prescriptive model (a model, possibly a mathematical/statistical model, is a representation of how the world works)**
3. **Deploy the model in an application to describe the current state, diagnose the cause of problems in the current state, predict the future state, or prescribe what needs to be done now based on the input**

The data science approach provides a different way to address the challenge of numerous variables. It requires significant amounts of historical data that is then given as input to existing algorithms (machine learning algorithms) e.g. k-Means clustering algorithm, Naïve Bayes, adaptive Bayes network, the Support Vector Machine algorithm, A Priori, Eclat, FP growth, amongst many other options. Here is a list of more algorithms: https://en.wikipedia.org/wiki/List_of_algorithms. This results in a model that can be used to process new and unforeseen data. The data scientist can then input the new data into the model (as opposed to an algorithm) which can then consider numerous variables simultaneously and produce a result.

Difference Between Business Intelligence and Data Science


These are business questions that can be answered by a data science approach as opposed to a traditional programming approach. This involves analysing information to produce knowledge (DIKAR model). Notice the number of parameters that must be considered simultaneously to answer each question.
Descriptive
• What percentage of the East African market does the company control now?
Diagnostic
• Why did the company's percentage of control in the North African market decrease by 20% over the last 5 years?
Predictive
• What percentage of the East African market will the company control next year?
Prescriptive
• What action can the company implement now to control at least 80% of the East African market before the next general elections in Kenya?
Value
Difficulty

The term "data science", as used in business, involves an interdisciplinary approach to make inferences from available business data. This data is usually in large quantities with hundreds or thousands of variables (columns) and complex. On the other hand, Business Intelligence (BI) helps to monitor the current state of business data to understand the historical performance of a business. While BI helps to interpret past data, data science can analyze past data to make predictions and prescriptions. BI is, therefore, mainly used for descriptive or diagnostic analytics in the context of a business whereas data science, which can handle many variables simultaneously, is mainly used for predictive or prescriptive analytics any science-related disciplines beyond the context of a business. Those who choose the BI option, as opposed to the networking option, in BBIT 4th year will learn more about Business Intelligence.

SQL as a Non-Procedural Programming Language

SQL is a non-procedural (declarative) programming language. This means that it specifies what is to be done rather than how (the procedure) to do it. A DBA, therefore, uses SQL as a tool to express what is desired in terms of what is known, e.g., if it is known that the database has a relation that stores client's data, then a DBA can specify that he/she wants a report on the list of clients that are being served by a specific sales representative.

This prevents the customization of instructions on how a task is to be executed, therefore, SQL needs to be supplemented by a procedural programming language that can then implement the traditional programming approach. Those interested in a recap of different types of computer programming languages can refer to the online version of the Britannica Encyclopedia via this link: https://www.britannica.com/technology/computer-programming-language Also, SQL does not offer an environment to implement machine learning algorithms that can be used to build models. **These factors limit SQL to being used to support Business Intelligence applications in a traditional programming approach as opposed to being used for data science in a data science approach.**

This lab manual focuses on creating, listing, deleting, and using stored routines (procedures and functions). The stored routines can eventually be used to retrieve data required for descriptive and diagnostic analytics in Business Intelligence (BBIT 4th year option). An even higher level of analytics (predictive and prescriptive analytics) can be implemented in data science based on data that can be retrieved using stored routines. Your ability to retrieve data using stored routines, therefore, forms a fundamental foundation for Business Intelligence (BI) and possibly data science in the future. This knowledge is required to develop business information systems at the management support layer.

Advantages of Procedures

(i) **Reduce network traffic:** Stored procedures help to reduce the network traffic between applications and the database system because instead of sending multiple lengthy SQL statements, applications can send only the name and parameters of stored procedures.

(ii) **Centralize business logic in the database:** You can use the stored procedures to implement business logic that is reusable by multiple applications. The stored procedures help reduce the efforts of duplicating the same logic in many applications.

(iii) **Make the database more secure:** The database administrator can grant appropriate privileges to applications to allow them to access specific stored procedures without giving any privileges on the underlying tables. This implies that the application has a personalized account to connect to the database (not the root account).

SUPPOSE that the business strategy committee at Classic Models identifies an **attractive market opportunity** in East Africa. This committee then retrieves historical data to conduct a SWOT analysis, that is, an analysis which aims to identify the Strengths, Weaknesses, Opportunities, and Threats that a business faces. The results of this analysis indicate that the business has the right **resources, competencies,** and **capabilities** to venture into the East African market. They also note that the business can make the right **compromises** that can enable it to venture into this new market. A decision is made by the Board of Directors (BOD) as informed by the business strategy committee. The decision is to **execute** the entry of Classic Models into the East African market with the **discipline** and determination to succeed. They decide to setup their East African office in Nairobi.

STEP 5. Create a record for the Nairobi office

Execute the following command to create a record for the Nairobi office:

```
INSERT
      INTO
      offices (`officeCode`,
      `city`,
      `phone`,
      `addressLine1`,
      `addressLine2`,
      `country`,
      `postalCode`,
      `territory`)
VALUES ('8',
'Nairobi',
'+254 720 123 456',
'16 5th Ngong Avenue, Suite 11',
'Nairobi, NBO 00202',
'Kenya',
'00202',
'EMEA');
```

STEP 6. Create records for 5 sales representatives who work in the Nairobi office

They then recruit 5 sales representatives to work in the Nairobi Office. Execute the following command to create records of the 5 employees:

```
INSERT INTO `employees`
    (`employeeNumber`, `lastName`, `firstName`, `extension`, `email`, `officeCode`, `reportsTo`, `jobTitle`)
VALUES
    ('1703', 'Kiprono', 'John', 'x211', 'jkiprono@classicmodelcars.com', '8', '1056', 'Sales Rep'),
    ('1704', 'Naliaka', 'Mary', 'x212', 'mnaliaka@classicmodelcars.com', '8', '1056', 'Sales Rep'),
    ('1705', 'Mogaka', 'Andrew', 'x213', 'amogaka@classicmodelcars.com', '8', '1056', 'Sales Rep'),
    ('1706', 'Shanyisa', 'Esther', 'x214', 'eshanyisa@classicmodelcars.com', '8', '1056', 'Sales Rep'),
    ('1707', 'Oyier', 'Joshua', 'x215', 'joyier@classicmodelcars.com', '8', '1056', 'Sales Rep');
```

STEP 7. Create a relation to store the salaries of the employees in the Nairobi office

The Nairobi office requires a place to store the remuneration amount (salary) of each of its 5 employees. Create a table called "employees_salary" to record the amount of salary for each employee. This can be done by executing the following command:

```
CREATE TABLE `employees_salary` (
  `employeeNumber` int NOT NULL,
  `employees_salary_amount` decimal(9,2) NOT NULL DEFAULT '0.00',
  PRIMARY KEY (`employeeNumber`),
  UNIQUE KEY `IDX_employeeNumber_UNIQUE` (`employeeNumber`),
  CONSTRAINT `FK_1_employees_salary_TO_1_employees` FOREIGN KEY (`employeeNumber`) REFERENCES `employees`
(`employeeNumber`)
) ENGINE = InnoDB COMMENT = 'Records the salary of each employee'
```

STEP 8. Record the salary of each employee working in the Nairobi office

Execute the following statements to record the salary for each of the 5 employees based in Nairobi:

```
INSERT INTO employees_salary (employeeNumber, employees_salary_amount)
VALUES
    ('1703', '11000'),
    ('1704', '20000'),
    ('1705', '34000'),
    ('1706', '38500'),
    ('1707', '65000');
```

SUPPOSE that the Payroll Office has requested for your services as a Database Administrator to compute the individual income tax for each employee. This computation is based on the pre-determined Pay-As-You-Earn (PAYE) formula that is already being used manually by the Payroll Office. PAYE is the process by which a government (through its Revenue Authority) collects employees' income tax directly from the employer. It is the employer's statutory duty to deduct income tax from the remuneration of all employees who earn a monthly salary of more than KES. 11,135 in Kenya. In this scenario, the multi-national company (classic models) deducts income tax depending on the government rules, e.g. the residential employees (citizens) who work in Kenya may pay a different income tax from the residential employees who work in say, France or in the company's Japan office. The Government of Kenya stipulates that the following tax rates should be applied:

Monthly Taxable Income Range (KES)	**Taxable Income (KES)** *(i.e. the total amount in this income range)*	**Tax Rate (%)**	**Income Tax Charged**
0 - 12,298.99	$12{,}298 - 0 = 12{,}298.99$	10	$10\% \times 12{,}298.99$
12,299 - 23,885.99	$23{,}885.99 - 12{,}299 = 11{,}586.99$	15	$15\% \times 11{,}586.99$
23,886 - 35,472.99	$35{,}472.99 - 28{,}886 = 11{,}586.99$	20	$20\% \times 11{,}586.99$
35,473 - 47,059.99	$47{,}059.99 - 35{,}473 = 11{,}586.99$	25	$25\% \times 11{,}586.99$
47,060 and above	$< total\ taxable\ income > - 47{,}060 = x$	30	$30\% \times x$

For example, if an employee's taxable income is KES. 38,500.00 per month, then the total income tax the employee is expected to pay to the government is KES. 6,042.00 per month. This is computed as follows:

Monthly Taxable Income Range (KES)	**Taxable Income (KES)**	**Tax Rate (%)**	**Income Tax Charged**
0 - 12,298.99	12,298.99	10	1,229.80
12,299 - 23,885.99	11,586.99	15	1,738.05
23,886 - 35,472.99	11,586.99	20	2,317.40
35,473 - 47,059.99	3,027	25	756.75
47,060 and above	-	30	-
TOTAL	**≈38,500**		**6,042.00**

Through meetings between the Payroll Office and the IT Department's database team, the following algorithm has been stipulated as the formula used to calculate an individual employee's PAYE income tax in the Nairobi office.

Algorithm 1: PAYE (income tax) for employees working in the Kenyan office

```
function PAYE($employeeID$)
    $sal$ = salary of $employeeID$
    $tax = 0.00$
    if $sal > 11,135$ then
        if $sal \geq 12,298.99$ then
            $tax = tax + (0.1 * 12,298.99)$
        else if $sal < 12,298.99$ then
            $tax = tax + (0.1 * sal)$
        end if
        if $sal \geq 23,885.99$ then
            $tax = tax + (0.15 * 11,586.99)$
        else if $sal < 23,885.99$ and $sal > 12,299$ then
            $tax = tax + (0.15 * (sal - 12,299))$
        end if
        if $sal \geq 35,472.99$ then
            $tax = tax + (0.2 * 11,586.99)$
        else if $sal < 35,472.99$ and $sal > 23,886$ then
            $tax = tax + (0.2 * (sal - 23,886))$
        end if
        if $sal \geq 47,059.99$ then
            $tax = tax + (0.25 * 11,586.99)$
        else if $sal < 47,059.99$ and $sal > 35,473$ then
            $tax = tax + (0.25 * (sal - 35,473))$
        end if
        if $sal \geq 47,060$ then
            $tax = tax + (0.3 * (sal - 47,060))$
        end if
        Return TOTAL monthly PAYE income tax of $employeeID$ as $tax$
    end if
end function
```

STEP 9. Create a procedure called "PROC_PAYE_Kenya" to implement this algorithm

The stored procedure should accept the employee's ID as input and provide output as the tax that the individual employee is required to pay. Execute the following command to create the procedure (use the CLI outlined in STEP 4 instead of DBeaver so that the delimiter is recognized):

```sql
DELIMITER $$
CREATE PROCEDURE `proc_PAYE_Kenya`(IN employeeID DOUBLE, OUT tax DOUBLE)
BEGIN
DECLARE sal DOUBLE;
SET tax = 0.00;
SELECT
    employees_salary_amount
INTO sal FROM
    employees_salary
WHERE
    employeeNumber = employeeID;
if sal > 11135 THEN
            if sal >= 12289.99 THEN
                SET tax = tax + (0.1*12298.99);
            elseif sal < 12298.99 THEN
                SET tax = tax + (0.1 * sal);
            end if;
            if sal >= 23885.99 THEN
                SET tax = tax + (0.15*11586.99);
            elseif sal < 23885.99 AND sal > 12299 THEN
                SET tax = tax + (0.15 * (sal-12299));
            end if;
            if sal >= 35472.99 THEN
                SET tax = tax + (0.2*11586.99);
            elseif sal < 35472.99 AND sal > 23886 THEN
                SET tax = tax + (0.2 * (sal-23886));
            end if;
            if sal >= 47059.99 THEN
                SET tax = tax + (0.25*11586.99);
            elseif sal < 47059.99 AND sal > 35473 THEN
                SET tax = tax + (0.25 * (sal-35473));
            end if;
            if sal >= 47060 THEN
                SET tax = tax + (0.3*(sal - 47060));
            end if;
end if;
END$$
DELIMITER ;
```

- `DELIMITER $$` – Changes the delimiter symbol from the default semicolon (`;`) to two backslashes (`$$`). Once the procedure has been created, the delimiter is returned to a semicolon (`;`). Please refer to the "Lab Manual on Triggers" for a detailed explanation of delimiters.
- `` CREATE PROCEDURE `proc_PAYE_Kenya` `` – This forms part of a DDL statement (a sub-language of SQL) that allows you to define a database object called a procedure and to specify the name of the procedure. A DBA requires an account that has the "`CREATE ROUTINE`" privilege to create stored routines (both procedures and functions).

  Execute the following to view all the privileges that the account you have used to connect has:

  ```
  SHOW GRANTS FOR CURRENT_USER();
  ```

  ```
  Grants for student@%          |
  ----------------------------+
  GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER,
  SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT,
  CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE, CREATE
  ROLE, DROP ROLE ON *.* TO `student`@`%` WITH GRANT OPTION

  GRANT
  ALLOW_NONEXISTENT_DEFINER,APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,AUTHENTICATION_POLICY_ADMIN,BA
  CKUP_ADMIN,BINLOG_ADMIN,BINLOG_ENCRYPTION_ADMIN,CLONE_ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,FIREWALL_EXEMPT,
  FLUSH_OPTIMIZER_COSTS,FLUSH_STATUS,FLUSH_TABLES,FLUSH_USER_RESOURCES,GROUP_REPLICATION_ADMIN,GROUP_REPLICATION_STR
  EAM,INNODB_REDO_LOG_ARCHIVE,INNODB_REDO_LOG_ENABLE,PASSWORDLESS_USER_ADMIN,PERSIST_RO_VARIABLES_ADMIN,REPLICATION_
  APPLIER,REPLICATION_SLAVE_ADMIN,RESOURCE_GROUP_ADMIN,RESOURCE_GROUP_USER,ROLE_ADMIN,SENSITIVE_VARIABLES_OBSERVER,S
  ERVICE_CONNECTION_ADMIN,SESSION_VARIABLES_ADMIN,SET_ANY_DEFINER,SHOW_ROUTINE,SYSTEM_USER,SYSTEM_VARIABLES_ADMIN,TA
  BLE_ENCRYPTION_ADMIN,TELEMETRY_LOG_ADMIN,TRANSACTION_GTID_TAG,XA_RECOVER_ADMIN ON *.* TO `student`@`%` WITH GRANT
  OPTION
  ```

- **`BEGIN END block`** – The code in a procedure is placed between a BEGIN END block. The delimiter symbol is then placed after the END keyword to end the procedure.
- `(IN employeeID DOUBLE, OUT tax DOUBLE)` – A parameter can have one of three modes: IN, OUT, or INOUT. The syntax used to define a parameter is:

  ```
  [IN | OUT | INOUT] parameter_name datatype[(length)]
  ```

In this case, `employeeID` is an IN parameter of the type **`DOUBLE`** whereas `tax` is an OUT parameter of the type **`DOUBLE`**.

- **`IN parameter:`** IN is the default mode for parameters. An IN parameter is used to give input to a procedure. A procedure works on **a copy of** the IN parameter instead of on the original IN parameter. This means that the original value of an IN parameter is retained even after the procedure's execution ends.
- **`OUT parameter:`** An OUT parameter is used to receive output from a procedure. The value of an OUT parameter can be changed inside the procedure. The new value of the OUT parameter is then passed back as output to the calling program.
- **`INOUT parameter:`** A combination of IN and OUT parameters. This means that the calling program can pass an argument as input, and the procedure can modify the INOUT parameter, and pass the new value back to the calling program as output.

- **`DECLARE sal DOUBLE;`** – A variable is a named data object whose value can change during the execution of the procedure. Variables are used in procedures to hold immediate results. These variables are local to the stored procedure. Before using a variable, you must declare it. The syntax for declaring a variable in SQL is

  ```
  DECLARE variable_name datatype(size) [DEFAULT default_value];
  ```

- **`SET tax = 0.00;`** – The SET statement can be used to assign values to a variable once the variable has been declared. The syntax for the SET statement is:

  ```
  SET variable_name = value;
  ```

- **`SELECT employees_salary_amount INTO sal FROM employees_salary WHERE emp_ID = employeeID;`** – The SELECT INTO statement provides an alternative way of assigning values to a variable once the variable has been declared. In this case, the variable called **`sal`** is assigned the value of the employee's salary.

Variable Scope

A variable has its own scope that defines its lifetime. If you declare a variable inside a procedure, it will be out of scope when the END statement of the procedure is executed. When you declare a variable inside the BEGIN END block, it will be out of scope when the END statement of the BEGIN END block is executed. Being "out of scope" means that you can no longer access its value.

The format `@variable_name`, where the `variable_name` consists of alphanumeric characters, is used to create a user-defined variable outside a procedure and outside any BEGIN END block. A user-defined variable defined by one client is not visible to other clients. In other words, a user-defined variable is session-specific. You can, however, refer to the variable during the session, for example:

```
SET @tax = 0.00;
CALL proc_PAYE_Kenya(1703,@tax);
SELECT @tax;
```

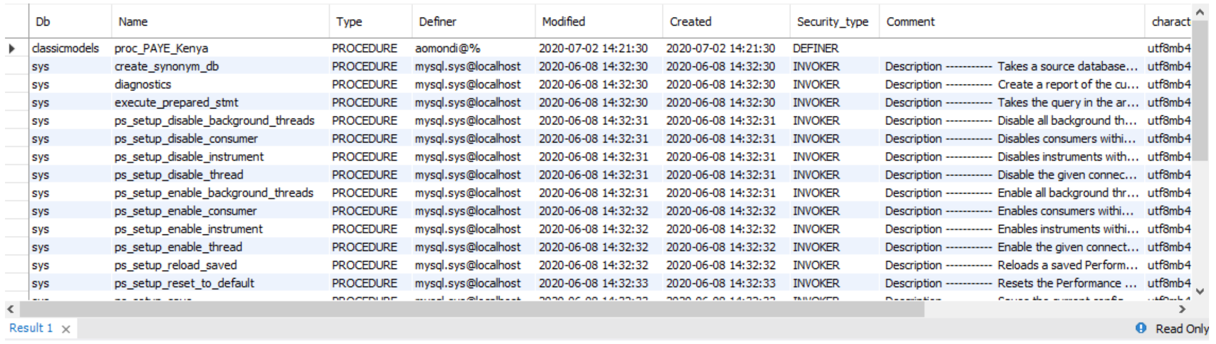
In this case, `SELECT @tax;` is used to display the value that is stored inside the variable `@tax`. The value will be maintained for as long as the user is logged in (i.e., during the session).

STEP 10. Show the created procedure

The SHOW PROCEDURE STATUS statement lists all the stored procedures in the database that you have a privilege to access. Execute the following command to confirm that the stored procedure has been created:

```
SHOW PROCEDURE STATUS;
```

You should get an output similar to (not identical to) the following:

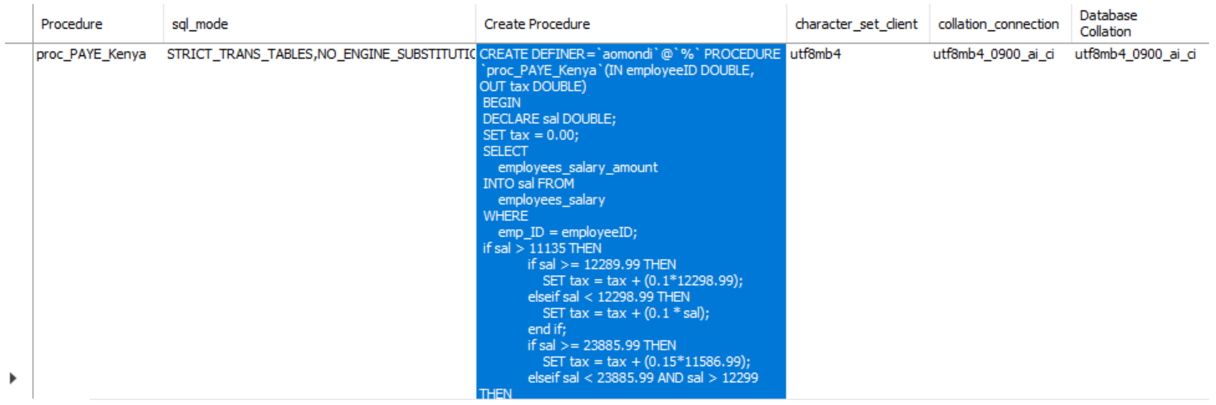


Db	Name	Type	Definer	Modified	Created	Security_type	Comment	charact
classicmodels	proc_PAYE_Kenya	PROCEDURE	aomondi@%	2020-07-02 14:21:30	2020-07-02 14:21:30	DEFINER		utf8mb4
sys	create_synonym_db	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:30	2020-06-08 14:32:30	INVOKER	Description ---------- Takes a source database...	utf8mb4
sys	diagnostics	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:30	2020-06-08 14:32:30	INVOKER	Description ---------- Create a report of the cu...	utf8mb4
sys	execute_prepared_stmt	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:30	2020-06-08 14:32:30	INVOKER	Description ---------- Takes the query in the ar...	utf8mb4
sys	ps_setup_disable_background_threads	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:31	2020-06-08 14:32:31	INVOKER	Description ---------- Disable all background th...	utf8mb4
sys	ps_setup_disable_consumer	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:31	2020-06-08 14:32:31	INVOKER	Description ---------- Disables consumers withi...	utf8mb4
sys	ps_setup_disable_instrument	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:31	2020-06-08 14:32:31	INVOKER	Description ---------- Disables instruments with...	utf8mb4
sys	ps_setup_disable_thread	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:31	2020-06-08 14:32:31	INVOKER	Description ---------- Disable the given connec...	utf8mb4
sys	ps_setup_enable_background_threads	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:31	2020-06-08 14:32:31	INVOKER	Description ---------- Enable all background thr...	utf8mb4
sys	ps_setup_enable_consumer	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:32	2020-06-08 14:32:32	INVOKER	Description ---------- Enables consumers withi...	utf8mb4
sys	ps_setup_enable_instrument	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:32	2020-06-08 14:32:32	INVOKER	Description ---------- Enables instruments withi...	utf8mb4
sys	ps_setup_enable_thread	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:32	2020-06-08 14:32:32	INVOKER	Description ---------- Enable the given connect...	utf8mb4
sys	ps_setup_reload_saved	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:32	2020-06-08 14:32:32	INVOKER	Description ---------- Reloads a saved Perform...	utf8mb4
sys	ps_setup_reset_to_default	PROCEDURE	mysql.sys@localhost	2020-06-08 14:32:33	2020-06-08 14:32:33	INVOKER	Description ---------- Resets the Performance ...	utf8mb4

The first row shows the procedure “proc_PAYE_Kenya” that is inside the “classicmodels” database.

```
SHOW CREATE PROCEDURE proc_PAYE_Kenya;
```

You should get the following output:



Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	Database Collation
proc_PAYE_Kenya	STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTIO	CREATE DEFINER=\`aomondi\`@\`%\` PROCEDURE \`proc_PAYE_Kenya\`(IN employeeID DOUBLE, OUT tax DOUBLE) BEGIN DECLARE sal DOUBLE; SET tax = 0.00; SELECT employees_salary_amount INTO sal FROM employees_salary WHERE emp_ID = employeeID; if sal > 11135 THEN if sal >= 12289.99 THEN SET tax = tax + (0.1*12298.99); elseif sal < 12298.99 THEN SET tax = tax + (0.1 * sal); end if; if sal >= 23885.99 THEN SET tax = tax + (0.15*11586.99); elseif sal < 23885.99 AND sal > 12299 THEN	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

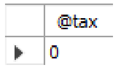
```
CREATE DEFINER=`aomondi`@`%` PROCEDURE
`proc_PAYE_Kenya`(IN employeeID DOUBLE,
OUT tax DOUBLE)
BEGIN
DECLARE sal DOUBLE;
SET tax = 0.00;
SELECT
   employees_salary_amount
INTO sal FROM
   employees_salary
WHERE
   emp_ID = employeeID;
if sal > 11135 THEN
         if sal >= 12289.99 THEN
            SET tax = tax + (0.1*12298.99);
         elseif sal < 12298.99 THEN
            SET tax = tax + (0.1 * sal);
         end if;
         if sal >= 23885.99 THEN
            SET tax = tax + (0.15*11586.99);
         elseif sal < 23885.99 AND sal > 12299
THEN
```

STEP 11. Call the stored procedure

The CALL statement allows you to invoke a stored procedure. The first time you invoke a stored procedure, the MySQL/MariaDB DBMS looks up the name of the procedure in the database catalogue, compiles the stored procedure’s code, places the compiled version in a memory area called the cache, and then executes the stored procedure. If you invoke the same stored procedure in the same session again, MySQL executes the stored procedure from the cache without having to recompile it.

Execute the following commands to call the procedure and get John Kiprono’s (employee number 1703) income tax:

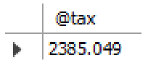
```
SET @tax = 0.00;
CALL proc_PAYE_Kenya(1703,@tax);
SELECT @tax;
```



```
@tax
0
```

Execute the following commands to call the procedure and get Mary Naliaka's (employee number 1704) income tax:

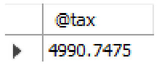
```
SET @tax = 0.00;
CALL proc_PAYE_Kenya(1704,@tax);
SELECT @tax;
```



```
@tax
2385.049
```

Execute the following commands to call the procedure and get Andrew Mogaka's (employee number 1705) income tax:

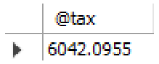
```
SET @tax = 0.00;
CALL proc_PAYE_Kenya(1705,@tax);
SELECT @tax;
```



```
@tax
4990.7475
```

Execute the following commands to call the procedure and get Esther Shanyisa's (employee number 1706) income tax:

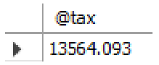
```
SET @tax = 0.00;
CALL proc_PAYE_Kenya(1706,@tax);
SELECT @tax;
```



```
@tax
6042.0955
```

Execute the following commands to call the procedure and get Joshua Oyier's (employee number 1707) income tax:

```
SET @tax = 0.00;
CALL proc_PAYE_Kenya(1707,@tax);
SELECT @tax;
```



```
@tax
13564.093
```

STEP 12. Create a function called "FUNC_PAYE_Kenya" to implement this algorithm

A few days later, the Payroll Office request for an easier way to get the income tax of all employees in the database using one command instead of a stored procedure that calculates the income tax for only one employee at a time. Create an SQL function called FUNC_PAYE_Kenya to compute the income tax of all the employees who work in the Nairobi office in Kenya. The function should be called only once to get the income tax for each of the 5 employees and its input should be the employee's salary. Execute the following command to create the function (use the CLI outlined in STEP 4 instead of DBeaver so that the delimiter is recognized):

```
DELIMITER $$
CREATE FUNCTION `FUNC_PAYE_Kenya`(sal DOUBLE) RETURNS DOUBLE
DETERMINISTIC
BEGIN
DECLARE tax DOUBLE;
SET tax = 0.00;

if sal > 11135 THEN
            if sal >= 12298.99 THEN
                SET tax = tax + (0.1*12298.99);
            elseif sal < 12298.99 THEN
                SET tax = tax + (0.1 * sal);
            end if;
            if sal >= 23885.99 THEN
                SET tax = tax + (0.15*11586.99);
            elseif sal < 23885.99 AND sal > 12299 THEN
                SET tax = tax + (0.15 * (sal-12299));
            end if;
            if sal >= 35472.99 THEN
                SET tax = tax + (0.2*11586.99);
            elseif sal < 35472.99 AND sal > 23886 THEN
                SET tax = tax + (0.2 * (sal-23886));
            end if;
            if sal >= 47059.99 THEN
                SET tax = tax + (0.25*11586.99);
            elseif sal < 47059.99 AND sal > 35473 THEN
                SET tax = tax + (0.25 * (sal-35473));
            end if;
            if sal >= 47060 THEN
                SET tax = tax + (0.3*(sal - 47060));
            end if;
end if;
RETURN tax;
END$$
DELIMITER ;
```

- `DETERMINISTIC` – A deterministic function always returns the same result for the same input parameters whereas a non-deterministic function returns different results for the same input parameters. In other words, you can classify a function as deterministic when you expect the output to always be the same as long as you give it the same input, e.g.
    - if f(x) = 2x then output will always be 2 x 5 = 10 when x = 5.
    - However, if g(x) = 2x + <current_date> then the output will be:
    - (2 x 5) + 12 = 22 when the date is 12th and x = 5.

    - And the value will be:
    - (2 x 5) + 13 = 23 when the date is 13th and x = 5.

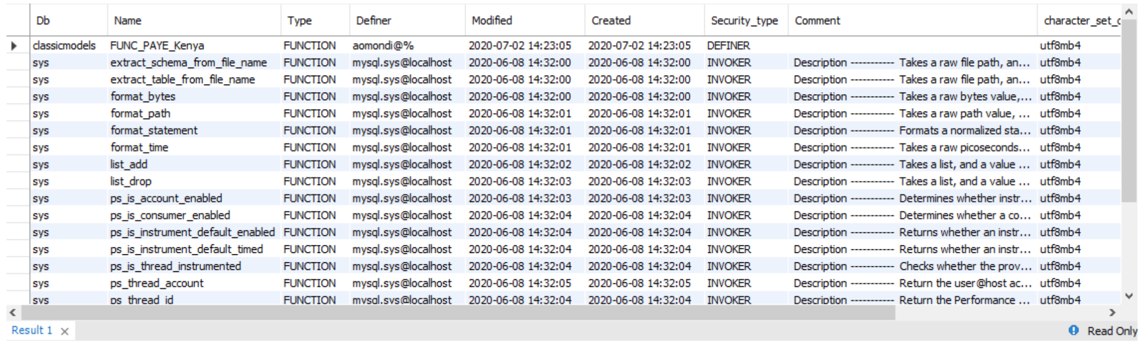
    - In this case, f(x) is a deterministic function and g(x) is a non-deterministic function. The non-deterministic functions can be used in cases where the function performs a calculation using a value that varies e.g., time or a dynamic variable read from a table in the database.

STEP 13. Show the created function

The SHOW FUNCTION STATUS statement lists all the functions in the database that you have a privilege to access. Execute the following command to confirm that the function has been created:

```
SHOW FUNCTION STATUS;
```

You should get an output similar to (not identical to) the following:

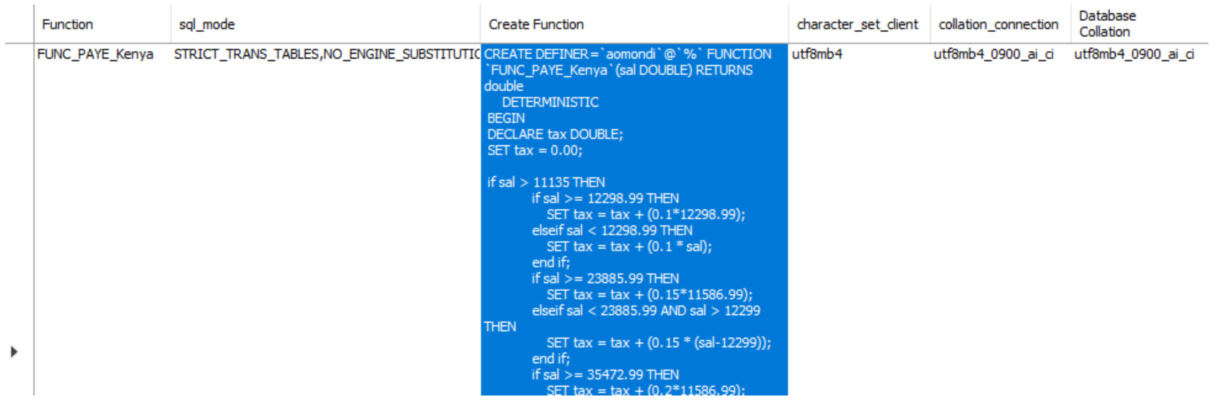


Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_c
classicmodels	FUNC_PAYE_Kenya	FUNCTION	aomondi@%	2020-07-02 14:23:05	2020-07-02 14:23:05	DEFINER		utf8mb4
sys	extract_schema_from_file_name	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:00	2020-06-08 14:32:00	INVOKER	Description ---------- Takes a raw file path, an...	utf8mb4
sys	extract_table_from_file_name	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:00	2020-06-08 14:32:00	INVOKER	Description ---------- Takes a raw file path, an...	utf8mb4
sys	format_bytes	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:00	2020-06-08 14:32:00	INVOKER	Description ---------- Takes a raw bytes value,...	utf8mb4
sys	format_path	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:01	2020-06-08 14:32:01	INVOKER	Description ---------- Takes a raw path value, ...	utf8mb4
sys	format_statement	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:01	2020-06-08 14:32:01	INVOKER	Description ---------- Formats a normalized sta...	utf8mb4
sys	format_time	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:01	2020-06-08 14:32:01	INVOKER	Description ---------- Takes a raw picoseconds...	utf8mb4
sys	list_add	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:02	2020-06-08 14:32:02	INVOKER	Description ---------- Takes a list, and a value ...	utf8mb4
sys	list_drop	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:03	2020-06-08 14:32:03	INVOKER	Description ---------- Takes a list, and a value ...	utf8mb4
sys	ps_is_account_enabled	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:03	2020-06-08 14:32:03	INVOKER	Description ---------- Determines whether instr...	utf8mb4
sys	ps_is_consumer_enabled	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:04	2020-06-08 14:32:04	INVOKER	Description ---------- Determines whether a co...	utf8mb4
sys	ps_is_instrument_default_enabled	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:04	2020-06-08 14:32:04	INVOKER	Description ---------- Returns whether an instr...	utf8mb4
sys	ps_is_instrument_default_timed	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:04	2020-06-08 14:32:04	INVOKER	Description ---------- Returns whether an instr...	utf8mb4
sys	ps_is_thread_instrumented	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:04	2020-06-08 14:32:04	INVOKER	Description ---------- Checks whether the prov...	utf8mb4
sys	ps_thread_account	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:05	2020-06-08 14:32:05	INVOKER	Description ---------- Return the user@host ac...	utf8mb4
sys	ps_thread_id	FUNCTION	mysql.sys@localhost	2020-06-08 14:32:04	2020-06-08 14:32:04	INVOKER	Description ---------- Return the Performance ...	utf8mb4

The first row shows the function "FUNC_PAYE_Kenya" that is inside the "classicmodels" database.

```
SHOW CREATE FUNCTION FUNC_PAYE_Kenya;
```

You should get the following output:



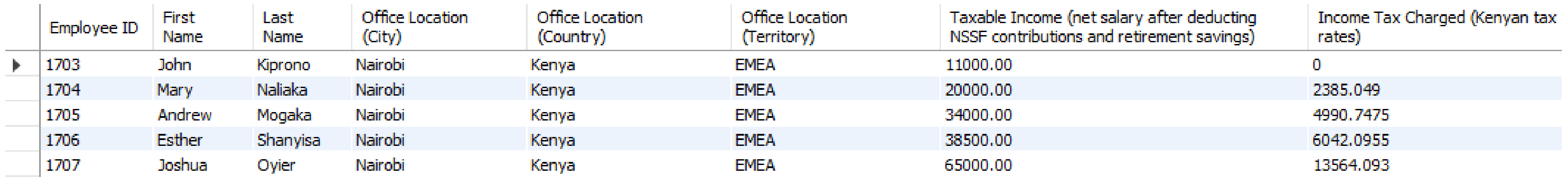
Function	sql_mode	Create Function	character_set_client	collation_connection	Database Collation
FUNC_PAYE_Kenya	STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTIC	CREATE DEFINER=\`aomondi\`@\`%\` FUNCTION \`FUNC_PAYE_Kenya\`(sal DOUBLE) RETURNS double DETERMINISTIC BEGIN DECLARE tax DOUBLE; SET tax = 0.00; if sal > 11135 THEN if sal >= 12298.99 THEN SET tax = tax + (0.1*12298.99); elseif sal < 12298.99 THEN SET tax = tax + (0.1 * sal); end if; if sal >= 23885.99 THEN SET tax = tax + (0.15*11586.99); elseif sal < 23885.99 AND sal > 12299 THEN SET tax = tax + (0.15 * (sal-12299)); end if; if sal >= 35472.99 THEN SET tax = tax + (0.2*11586.99);	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

STEP 14. Call the function

Execute the following command to call the function and get the income tax for all the employees who work in the Nairobi office in Kenya:

```
SELECT
       employees_salary.employeeNumber AS 'Employee ID',
       employees.firstName AS 'First Name',
       employees.lastName AS 'Last Name',
       offices.city AS 'Office Location (City)',
       offices.country AS 'Office Location (Country)',
       offices.territory AS 'Office Location (Territory)',
       employees_salary.employees_salary_amount AS 'Taxable Income (net salary after deducting NSSF contributions and retirement savings)',
       FUNC_PAYE_KENYA(employees_salary.employees_salary_amount) AS 'Income Tax Charged (Kenyan tax rates)'
FROM
       employees_salary
JOIN
    employees ON
       employees_salary.employeeNumber = employees.employeeNumber
JOIN
    offices ON
       employees.officeCode = offices.officeCode
WHERE
       offices.officeCode = 8;
```

You should get the following result set as the output:



Employee ID	First Name	Last Name	Office Location (City)	Office Location (Country)	Office Location (Territory)	Taxable Income (net salary after deducting NSSF contributions and retirement savings)	Income Tax Charged (Kenyan tax rates)
1703	John	Kiprono	Nairobi	Kenya	EMEA	11000.00	0
1704	Mary	Naliaka	Nairobi	Kenya	EMEA	20000.00	2385.049
1705	Andrew	Mogaka	Nairobi	Kenya	EMEA	34000.00	4990.7475
1706	Esther	Shanyisa	Nairobi	Kenya	EMEA	38500.00	6042.0955
1707	Joshua	Oyier	Nairobi	Kenya	EMEA	65000.00	13564.093

Views

STEP 15. Create a view called "VIEW_PAYE_Kenya" to implement this algorithm

Suppose that the Payroll Office is still not yet satisfied and has requested for an even simpler way to view the income tax of each of the employees working in the Nairobi office in Kenya. The database team finally decides to create for them a simple view which they can invoke by executing the following statement only: `SELECT * FROM VIEW_PAYE_Kenya;`

The result should have the following format. Notice the change of name in the 7th column which is titled "Taxable Income" instead of "Taxable Income (net salary after deducting NSSF contributions and retirement savings)".

VIEW_PAYE_Kenya

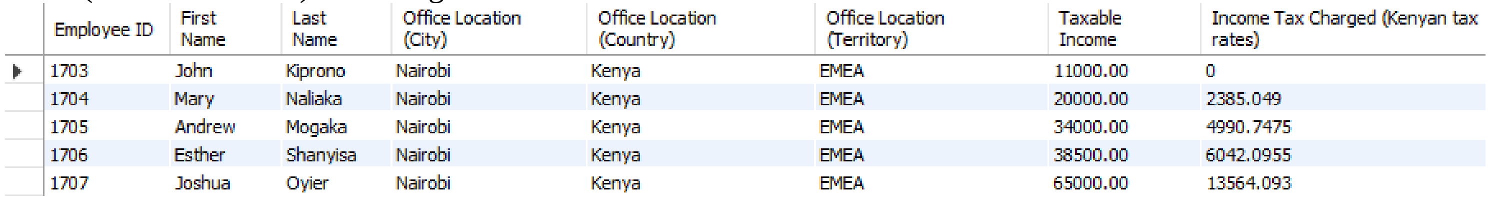
Employee ID	First Name	Last Name	Office Location (City)	Office Location (Territory)	Taxable Income	Income Tax Charged (Kenyan tax rates)
1703			*Nairobi*	*Kenya*		
1704			*Nairobi*	*Kenya*		
⋮			⋮	⋮		
1707			*Nairobi*	*Kenya*		

Execute the following command to create a view called “VIEW_PAYE_Kenya”:
```
CREATE VIEW `VIEW_PAYE_Kenya` AS
    SELECT
      employees_salary.employeeNumber AS 'Employee ID',
      employees.firstName AS 'First Name',
      employees.lastName AS 'Last Name',
      offices.city AS 'Office Location (City)',
      offices.country AS 'Office Location (Country)',
      offices.territory AS 'Office Location (Territory)',
      employees_salary.employees_salary_amount AS 'Taxable Income',
      FUNC_PAYE_KENYA(employees_salary.employees_salary_amount) AS 'Income Tax Charged (Kenyan tax rates)'
FROM
      employees_salary
JOIN
        employees ON
      employees_salary.employeeNumber = employees.employeeNumber
JOIN
        offices ON
      employees.officeCode = offices.officeCode
WHERE
      offices.officeCode = 8;
```

STEP 16. Call the view

Execute the following command to make use of the view:
```
SELECT * FROM VIEW_PAYE_Kenya;
```

This should give you a simpler way of retrieving the same data as you retrieved in STEP 14. The only difference is column 7 which has a shorter name (Taxable Income) according to the definition of the view.



Employee ID	First Name	Last Name	Office Location (City)	Office Location (Country)	Office Location (Territory)	Taxable Income	Income Tax Charged (Kenyan tax rates)
1703	John	Kiprono	Nairobi	Kenya	EMEA	11000.00	0
1704	Mary	Naliaka	Nairobi	Kenya	EMEA	20000.00	2385.049
1705	Andrew	Mogaka	Nairobi	Kenya	EMEA	34000.00	4990.7475
1706	Esther	Shanyisa	Nairobi	Kenya	EMEA	38500.00	6042.0955
1707	Joshua	Oyier	Nairobi	Kenya	EMEA	65000.00	13564.093

CURSORS and LOOPS
SUPPOSE that the marketing and sales primary function of the business' value chain requires help from the IT department to support their business process.

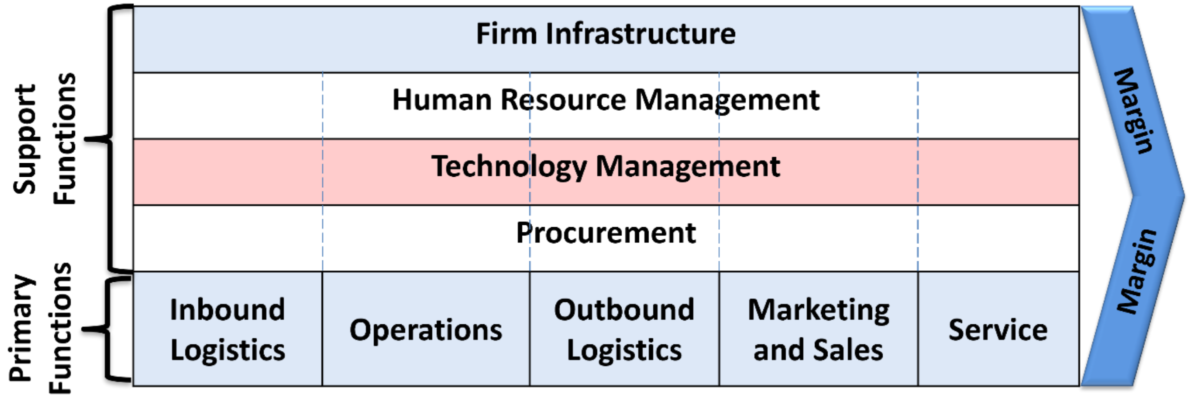


Figure 1: Porter's value chain diagram
Adapted from Johnson et al. (2008)

This will involve knowing (application of **technologies** to support access to the right **information**: notice the acronym "IT" which stands for "Information Technology") which clients have brought in the highest sales revenue into the business. This information will enable them to reward the top 10 clients in order to maintain their loyalty to the business.

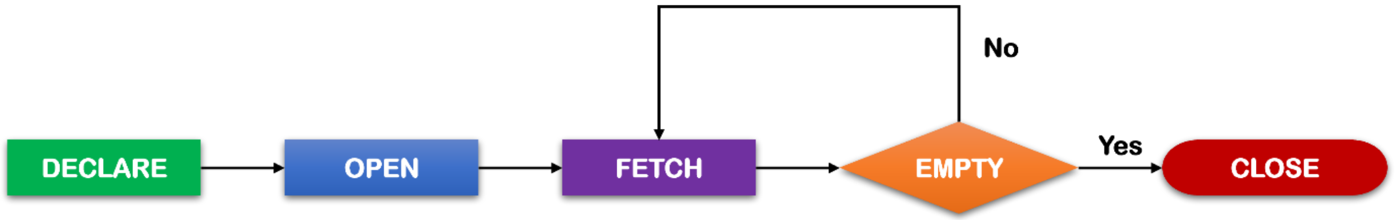
According to the marketing and sales department's requirements (not the DBA's/IT's requirements):
The marketing and sales team requires the report to show a textual (not tabular) list of the top 10 customers by displaying the information in the following manner: the rank of the customer (starting from number 10 to number 1) and in parenthesis, the name of the customer, followed by the customer's number in parenthesis, followed by the total amount they have brought into the business as sales revenue. Each record in the textual report should start on a new line. The amount should also use a comma for the thousand-separator, and it should have 2 decimal places e.g., 2,004.38 and not 2004.38.

STEP 17. Create a procedure to retrieve the information in the format required

Execute the following command to create the procedure (use the CLI outlined in STEP 4 instead of DBeaver so that the delimiter is recognized):

```
DELIMITER $$
CREATE PROCEDURE `proc_top10_customers`()

BEGIN

    DECLARE singleCustomerDetails TEXT DEFAULT "";
    DECLARE listOfTop10Customers TEXT DEFAULT "";
    DECLARE customerRank INTEGER DEFAULT 1;
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE CURSOR_listOfCustomers CURSOR FOR
    SELECT
        CONCAT(customers.customerName,
                ' [',
                payments.customerNumber,
                '] ',
                FORMAT(SUM(amount), 2))
    FROM
        payments
            INNER JOIN
        customers ON payments.customerNumber = customers.customerNumber
    GROUP BY payments.customerNumber
    ORDER BY SUM(amount) DESC
    LIMIT 0, 10;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

    OPEN CURSOR_listOfCustomers;

    LOOP_PopulateList: LOOP
        FETCH CURSOR_listOfCustomers INTO singleCustomerDetails;
        IF finished = 1 THEN
            LEAVE LOOP_PopulateList;
        END IF;
        SET listOfTop10Customers = CONCAT("(", customerRank, ") ", singleCustomerDetails, CHAR(10), listOfTop10Customers);
        SET customerRank = customerRank + 1;
    END LOOP LOOP_PopulateList;

    CLOSE CURSOR_listOfCustomers;

    SELECT listOfTop10Customers AS "List of Top 10 Customers";

    END$$
DELIMITER ;
```

- `DECLARE singleCustomerDetails TEXT DEFAULT "";` It is good practice to set a value after declaring a variable. This can be done not only by using the `SET` command but by also using the `DEFAULT` command.
- A cursor enables you to iterate through each row of a result set. This in turn supports the processing of one row at a time.
    - Cursors in MySQL and MariaDB are "**read-only**". This means that you cannot update the data retrieved from the original table **by using the cursor**.
    - Cursors in MySQL and MariaDB are also "**non-scrollable**" which means that you can only fetch rows in the order determined by the SELECT statement. Subsequently, you cannot skip rows or jump to a specific row in the result set.
    - Cursors in MySQL and MariaDB are also "asensitive" which means that they point to the actual data such that if the actual data is updated while using the cursor, then the cursor will use the updated version of the data. The opposite of this is an "insensitive" cursor which creates a temporary copy of the actual data and then points to this temporary copy.
- `DECLARE CURSOR_listOfCustomers CURSOR FOR <SELECT statement>;` – This is the syntax for creating a cursor. In this case, the cursor is called `CURSOR_listOfCustomers`. Notice the use of the prefix "`CURSOR_`" when naming database objects that are of the type "cursor". The declaration must be followed by a SELECT statement which will generate the result set. Note that the declaration of a CURSOR must be done **after** all the variables have been declared.
- The next step after declaring a CURSOR is to open the CURSOR, then to FETCH rows in the result set using the cursor, and lastly, to CLOSE the cursor once you are done using it i.e.:



- `OPEN CURSOR_listOfCustomers;` The syntax to open a cursor requires that the keyword "`OPEN`" is followed by the name of the cursor.
- `FETCH CURSOR_listOfCustomers INTO singleCustomerDetails;` The FETCH statement is then used to retrieve the next row pointed by the cursor and to move the cursor to the next row in the result set. In this scenario, the result set being pointed at by the cursor is placed inside the variable that was declared and set earlier called "`singleCustomerDetails`".
- `DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;` Handling an error can involve either a continuation of the execution of the current code block (syntax: `DECLARE CONTINUE`) or an exit of the current code block (syntax `DECLARE EXIT`). This is then later followed by an appropriate error message that informs the user of what happened, why it happened, and what they can do next.

- `DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;` "`NOT FOUND`" represents the condition that the handler is designed to handle. This can be stated as an error number, an SQLSTATE value, or, in this case, a named condition. The named condition provides a user-friendly way of specifying the error number or SQLSTATE. The common named condition associated with a CURSOR is "NOT FOUND".
- `DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;` This statement implies that if the "NOT FOUND" error is encountered, the value of the variable "`finished`" should be set to 1 and the execution of the current code block should continue. Additional examples of how to use a HANDLER can be found here.
- `LOOP_PopulateList: LOOP` Loops can be used to iterate through a section of code. Loops can optionally have names, which in this case is "`LOOP_PopulateList`". The name is followed by a full colon and then the keyword "`LOOP`" as per the syntax.
- `END LOOP LOOP_PopulateList;` Marks the end of the loop called "`LOOP_PopulateList;`".
- `LEAVE LOOP_PopulateList;` It is also possible to exit the loop before the END LOOP statement is reached. This is done using the LEAVE keyword.
- `CLOSE CURSOR_listOfCustomers;` Once you are done using the CURSOR, the CLOSE statement can be used to close the cursor.
- `SET listOfTop10Customers = CONCAT("(", customerRank, ") ", singleCustomerDetails, CHAR(10), listOfTop10Customers);` The "`CHAR()`" function supports the conversion of ASCII codes to character values. The following ASCII codes can be used to get a new line:

Value	Char	Description
10	LF	Line Feed (similar to a "new line" symbol "\n")
13	CR	Carriage Return (similar to a "return" symbol "\r")

STEP 18. Generate the report for the marketing and sales department

Execute the following command to generate the required report by calling the procedure. Note that the procedure has a SELECT statement inside it to display the result.

```
CALL `proc_top10_customers`();
```

You should get the following output:

```
(10) Saveley & Henriot, Co. [146] 130,305.35
(9) Corporate Gift Ideas Co. [321] 132,340.78
(8) Anna's Decorations, Ltd [276] 137,034.22
(7) AV Stores, Co. [187] 148,410.09
(6) Down Under Souveniers, Inc [323] 154,622.08
(5) Dragon Souveniers, Ltd. [148] 156,251.03
(4) Muscle Machine Inc [151] 177,913.95
(3) Australian Collectors, Co. [114] 180,585.07
(2) Mini Gifts Distributors Ltd. [124] 584,188.24
(1) Euro+ Shopping Channel [141] 715,738.98
```

Lab Submission Requirements

Please Note:

Use your creativity to create 1 procedure, 1 function, and 1 view to each retrieve data from multiple tables in the database. This data should be processed into information. The names (case-sensitive) should be as follows:

Procedure:	PROC_LAB5
Function:	FUNC_LAB5
View:	VIEW_LAB5

Save your code in the file named “`lab_submission.sql`”

Further Reading

Computer programming language. (2019). In *Encyclopedia Britannica*. Encyclopædia Britannica, Inc. https://www.britannica.com/technology/computer-programming-language

MySQL Stored Function By Practical Examples. (n.d.). *MySQL Tutorial*. Retrieved 2 July 2020, from https://www.mysqltutorial.org/mysql-stored-function/

MySQL Stored Procedure Tutorial. (n.d.). *MySQL Tutorial*. Retrieved 2 July 2020, from https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx/

MySQL Views. (n.d.). Retrieved 2 July 2020, from https://www.mysqltutorial.org/mysql-views-tutorial.aspx

Oracle. (2020). *MySQL :: MySQL 8.0 Reference Manual: 24.2 Using Stored Routines*. https://dev.mysql.com/doc/refman/8.0/en/stored-routines.html

Oracle. (2020). *MySQL :: MySQL 8.0 Reference Manual: 24.5 Using Views*. https://dev.mysql.com/doc/refman/8.0/en/views.html