MySQL Essentials

SAHITI KAPPAGANTULA
PRODUCT ASSOCIATE, OORWIN

MySQL is the standard query language used by tech professionals to handle multiple databases. Being one of the most popular open-source database management systems, MySQL ensures high security, availability, and ease of use. This Refcard will help you understand the fundamental concepts of MySQL, including core features, common applications, installation steps, and more.

But before we dive into MySQL, let's review the basics of databases and database management systems.

OVERVIEW OF DATABASES AND DBMSS

"Data" originated from the Latin word "datum," meaning something given, in the mid 18th century. It is the plural form of *datum*, meaning facts collected about something for further analysis. These chunks of data are stored in storage systems called "databases."

Databases are like vessels in which you can store data. To pick the chunks of data and manage them, we often require a database management system (DBMS).

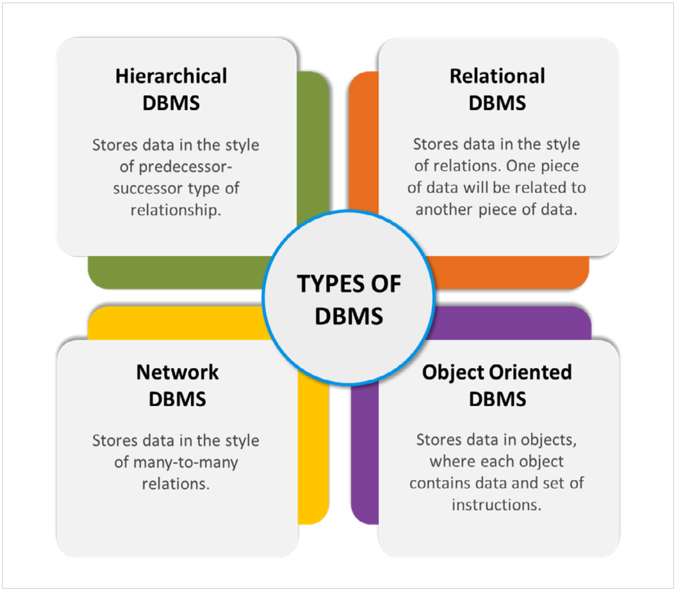
TYPES OF DBMSs

Database management systems are software applications used to handle databases. So, you can understand that a DBMS serves as an interface between the end user and the database(s). It takes inputs from the end user, performs actions on the databases, and displays the results.

There are four types of database management systems present in the market:

SEE FIGURE 1 IN THE NEXT COLUMN

Figure 1


Hierarchical DBMS
Stores data in the style of predecessor-successor type of relationship.
Relational DBMS
Stores data in the style of relations. One piece of data will be related to another piece of data.
TYPES OF DBMS
Network DBMS
Stores data in the style of many-to-many relations.
Object Oriented DBMS
Stores data in objects, where each object contains data and set of instructions.

WHAT IS SQL?

SQL, also known as the Structured Query Language, is the standard language used to deal with relational databases. Developed by Donald Chamberlin and Raymond F Boyce at IBM in the early 1970s, SQL allows users to access, manipulate, and store data in databases.

In 1986, SQL became the standard of the American National Standards Institute (ANSI), followed by the International Organization for Standardization (ISO) in 1987.

Today, SQL is used worldwide in various versions to insert, delete, update, and retrieve records. The most popular relational database — MySQL — uses the Structured Query Language to perform actions.

Let's further dive into MySQL, what it is, core features, and more.

ABOUT MYSQL

In this section of the Refcard, we will discuss the following basics of MySQL:

- What is MySQL
- MySQL features
- Applications of MySQL
- MySQL installation steps

WHAT IS MYSQL?

MySQL is the most popular open-source relational database management system developed, supported, and distributed by the Oracle corporation. The MySQL database server supports multi-user access and is very easy, fast, and reliable. It follows the client-server architecture and supports multiple client programs, back ends, etc. through multithreading.

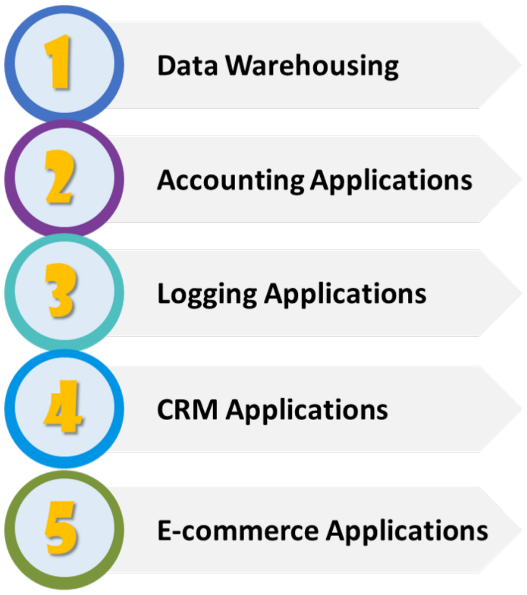
MYSQL FEATURES

- **Open source and scalable** – Supports multiple platforms and continuous scaling through replication, clustering, and/or sharding to store large databases.
- **High data security** — Protects your data using authentication plugins, SSL protocol, and data maskings to prevent cyber attacks.
- **Low cost and no downtime** – Comes with low license and hardware costs along with no downtime guaranteed.
- **Ease of availability and management** – Executes coordinator and worker configurations and allows users to schedule tasks.
- **High performance and transactional support** – Supports fast load utilities with table index partitioning and distinct memory caches. Also, it follows the ACID property to allow multi-version support.

APPLICATIONS OF MYSQL

Being one of the most popular open-source database management systems, MySQL makes it easy for the users to create and use database-driven applications. Popular applications of MySQL are shown below:

Figure 2



MYSQL INSTALLATION

MySQL can be installed on Linux using the following repositories:

- **Yum Repository**
- **MySQL SLES Repository**
- **MySQL APT Repository**

MYSQL INSTALLATION ON LINUX USING THE YUM REPOSITORY

With the MySQL Yum repository, you can install the MySQL client, MySQL server, MySQL router, MySQL workbench, MySQL utilities, MySQL shell, etc., on the Red Hat Enterprise Linux, CentOS, Oracle Linux, and Fedora.

1. Navigate to **MySQL Yum repository** in the URL: https://dev.mysql.com/downloads/repo/yum/
2. **Download the repository setup package** based on your platform
3. Now **install the downloaded package** by mentioning the command: `sudo yum install platform-and-version-specific-package-name.rpm`. Example: For an EL6 based system: `sudo yum install mysql80-community-release-el6-{version-number}.noarch.rpm`
4. To **check whether the MySQL Yum repository is added** or not, use the following command: `yum repolist enabled | grep "mysql.*-community.*"`.

In case you wish to go by the default MySQL series, skip the steps below.

Note: When you use the Yum repository, the latest MySQL series is selected by default. If you wish to choose another version, then complete the following steps:

a. Check the subrepositories currently enabled or disabled in the Yum repository using: `yum repolist all | grep mysql`

b. Disable the current MySQL series by mentioning its name: `sudo yum-config-manager --disable mysql57-community`

c. Enable the series you wish to use by mentioning its name: `sudo yum-config-manager --enable mysql80-community`

d. Verify whether the correct subrepositories have been enabled or not using: `yum repolist enabled | grep mysql`

e. Next, you must disable the MySQL module and make the MySQL repository packages visible: `sudo yum module disable mysql`

f. Now install MySQL: `sudo yum install mysql-community-server`. This will install the MySQL server, MySQL client, shared client libraries, etc.

5. **Start the MySQL Server:** `systemctl start mysqld` and check its status: `systemctl status mysqld`
   - At the initial startup of the server, the server is initialized, and the SSL certificate and key files are generated in the directory.
   - A superuser account `'root'@'localhost'` is created
   - The password validation component is installed
   - You can view all the MySQL components available from the MySQL Yum repository: `sudo yum --disablerepo=\* --enablerepo='mysql*-community*' list available`
   - Install any packages by replacing package-name with name of the package: `sudo yum install package-name.` Example: `sudo yum install mysql-workbench-community`

Now let's install MySQL using the SLES repository.

MYSQL INSTALLATION ON LINUX USING THE SLES REPOSITORY

With the MySQL SLES repository, you can install the MySQL components on SUSE Linux Enterprise Server (SLES).

1. Navigate to **MySQL SLES repository** in the URL: https://dev.mysql.com/downloads/repo/suse/.
2. **Download the repository setup package** based on the platform
3. Now **install the downloaded package** by mentioning the command: `sudo rpm -Uvh package-name.rpm`. Example: For an SLES 15 based system, use: `sudo rpm -Uvh mysql80-community-release-sl15-#.noarch.rpm`
4. **Import the MySQL GnuPG key** to check the signatures of the downloaded packages from the SLES repository: `sudo rpm --import /etc/RPM-GPG-KEY-mysql-2022`. In case you wish to go by the default MySQL series, skip the section below:

   a. When you use the SLES repository, the latest GA series is enabled by default and all other series are disabled

   b. View the enabled or disabled series using: `zypper repos | grep mysql.*community`

   c. You can enable another series by disabling the current series. Example: `sudo zypper modifyrepo -d mysql80-community`

   d. Now enable the desired series by mentioning its name: `sudo zypper modifyrepo -e mysql57-community`

   e. Verify whether the correct subrepositories have been enabled or not: `zypper repos -E | grep mysql.*community`

   f. Finally refresh the enabled subrepository: `sudo zypper refresh`

   g. Now install MySQL: `sudo zypper install mysql-community-server`. This will install the MySQL server along with other required packages

5. **Start the MySQL Server:** `systemctl start mysql` and check its status: `systemctl status mysql`

   a. At the initial startup of the server, the server is initialized, and the SSL certificate and key files are generated in the directory

   b. A superuser account `'root'@'localhost'` is created

   c. The password validation component is installed

   d. You must secure the MySQL installation for MySQL 5.6 version only by using the command: `mysql_secure_installation`

   e. You can list all the subrepositories in MySQL SLES repository: `zypper repos | grep mysql.*community`

   f. To list all the MySQL components in a subrepository, use the command: `zypper packages subrepo-name`

   g. Install any packages by replacing `package-name` with name of the package: `sudo zypper install package-name`. Example: `sudo zypper install mysql-community-bench`

Now let's install MySQL using the APT repository.

MYSQL INSTALLATION ON LINUX USING THE APT REPOSITORY

With the MySQL APT repository, you can install the MySQL components on the Debian and Ubuntu releases:

1. **Navigate to MySQL APT repository** in the URL: https://dev.mysql.com/downloads/repo/apt/
2. **Download the repository setup package** based on the platform
3. Now **install the downloaded package** by mentioning the command: `sudo dpkg -i /PATH/version-specific-package-name.deb`. Example: For the 0.8.22-1 version: `sudo dpkg -i mysql-apt-config_0.8.22-1_all.deb`.
4. You will be asked to choose the versions of MySQL components that you wish to install. **Select all of the components** and **click on OK** to complete the installation of the release package
5. Next **install the MySQL server**: `sudo apt-get install mysql-server` and check the status of the server: `systemctl status mysql`

Note: When you use the APT repository, the latest MySQL series is enabled by default and all other series are disabled. You can switch to another supported major release series by reconfiguring the installation: `sudo dpkg-reconfigure mysql-apt-config`:

1. **Update package information from the MySQL APT repository**: `sudo apt-get update`
2. Now **install the package** you wish by replacing the `package_name` with the package name: `sudo apt-get install package-name`. Example: `sudo apt-get install mysql-workbench-community`

With this, we've concluded the installation of MySQL on Linux using various repositories. You can also download MySQL on Windows and MacOS.

FUNDAMENTALS OF MYSQL

This section of the Refcard will cover fundamental concepts of MySQL such as:

- MySQL data types
- Common MySQL commands

MYSQL DATA TYPES

MySQL offers a varied range of data types segregated into various categories. The following table will brief you on each category's types, summary, and storage requirements.

DATA TYPE	CATEGORY	DESCRIPTION	STORAGE (BYTES)
`TINYINT(size)`	Numeric	Represents integer values. Supported range is: -128 to 127 signed values; 0 to 255 signed values.	1
`SMALLINT(size)`	Numeric	Represents integer values. Supported range is: -32768 to 32767 signed values; 0 to 65535 signed values.	2
`MEDIUMINT(size)`	Numeric	Represents integer values. Supported range is: -8388608 to 8388607 signed values; 0 to 16777215 signed values.	3
`INT(size)`	Numeric	Represents integer values. Supported range is: -2147483648 to 2147483647 signed values; 0 to 4294967295 signed values.	4
`BIGINT(size)`	Numeric	Represents integer values. Supported range is: -2 to 2^63 - 1 signed values; 0 to 2^64 -1 unsigned values	8
`FLOAT(precision, scale)`	Numeric	Stores approximate numeric data values.	4

TABLE CONTINUES ON THE NEXT PAGE

DATA TYPE	CATEGORY	DESCRIPTION	STORAGE (BYTES)
`DOUBLE(precision, scale)`	Numeric	Stores approximate numeric data values.	8
`DECIMAL(precision, scale)`	Numeric	Stores exact numeric data values.	Varies
`BIT(X)`	Numeric	Stores X- bit values from 1 to 64.	Maximum 4
`DATE`	Date and Time	Used for values containing only dates represented in "YYYY-MM-DD" format. The supported range is: "1000-01-01" to "9999-12-31."	3 as of MySQL 5.6.4
`TIME`	Date and Time	Used for values containing only time represented in "hh:mm:ss" format. The supported range is: - 838:59:59 to 838:59:59.	3 + fractional seconds storage as of MySQL 5.6.4
`DATETIME`	Date and Time	Used for values containing both date and time represented in "YYYY-MM-DD hh:mm:ss"format. The supported range is: "1000-01-01 00:00:00" to "9999-12-31 23:59:59."	5 + fractional seconds storage as of MySQL 5.6.4
`TIMESTAMP`	Date and Time	Used for values containing both date and time. The supported range is: "1970-01-01 00:00:01" UTC to "2038-01-19 03:14:07" UTC.	4 + fractional seconds storage as of MySQL 5.6.4
`YEAR`	Date and Time	Used to represent year values in YYYY format. The supported range is: 1901 to 2155 and 0000.	1 as of MySQL 5.6.4
`CHAR(length)`	String	Used to store fixed character strings with length ranging from 0 to 255.	(Length of characters * number of bytes) <=255
`VARCHAR(length)`	String	Used to store variable length strings with length ranging from 0 to 65535.	Length + 1 bytes if column require 0 - 255 bytes, else L + 2
`TINYTEXT`	String	Used to store strings with maximum 255 characters.	Length + 1 where Length < 2^8
`TEXT`	String	Used to store strings with maximum 65,535 characters.	Length + 2 where Length < 2^16
`MEDIUMTEXT`	String	Used to store strings with maximum 16,777,215 characters.	Length + 3 where Length < 2^24
`LONGTEXT`	String	Used to store strings with maximum 4,294,967,295 characters	Length + 4 where Length < 2^32
`TINYBLOB`	String	Used to store variable amounts of binary objects data. The maximum range is 255 bytes.	Length + 1 where Length < 2^8
`BLOB`	String	Used to store variable amounts of binary objects data. The maximum range is 65,535 bytes.	Length + 2 where Length < 2^16
`MEDIUMBLOB`	String	Used to store variable amounts of binary objects data. The maximum range is 16,777,215 bytes.	Length + 3 where Length < 2^24
`LONGBLOB`	String	Used to store variable amounts of binary objects data. The maximum range is 4,294,967,295 bytes.	Length + 4 where Length < 2^32
`ENUM('val1', 'val2', …)`	String	Stores list of values with the maximum as 65535	1 / 2 depending on the number of enum values (65,535 values maximum)
`SET('val1', 'val2', …)`	String	Stores list of values with the maximum as 64. It can also store duplicate values.	1 / 2 / 3 / 4 / 8 depending on the number of set members (64 members maximum)

These are some basic data types that you should know. There are two more advanced data types used in MySQL — spatial and JSON.

The spatial data type is used to store, analyze, and generate geographic features. The JSON data type enables easy access to data stored in JSON documents.

Now that you know the data types used in MySQL, let us explore the common MySQL commands with syntax and examples.

COMMON MYSQL COMMANDS WITH SYNTAX AND EXAMPLES

This section of the Refcard will help you understand the most used MySQL commands with examples.

COMMAND	DESCRIPTION	SYNTAX	EXAMPLE
CREATE USER	Creates users to access databases.	CREATE USER UserName IDENTIFIED BY 'Password';	CREATE USER Dzone123 IDENTIFIED BY 'DzoneRefcard@123';
CREATE DATABASE	Creates a new database.	CREATE SCHEMA DatabaseName;	CREATE SCHEMA InsuranceDetails;
CREATE TABLE	Creates a new table.	CREATE TABLE TableName (Column_1 Datatype, Column_2 Datatype, ...);	CREATE TABLE CarInsurance (PolicyID int, PolicyName varchar(255), EffectiveDate datetime, ExpiryDate datetime, PaymentOption varchar(255), Amount double, Status bool);
DROP USER	Delete users from the database server.	DROP USER UserName;	DROP USER Dzone123;
DROP DATABASE	Deletes the database along with the data present in it.	DROP SCHEMA DatabaseName;	DROP SCHEMA InsuranceDetails;
DROP TABLE	Deletes the table along with the data present in it.	DROP TABLE TableName;	DROP TABLE CarInsurance;

SHOW USER	Displays a list of users who have access to the database.	SELECT USER FROM MYSQL.
		DatabaseName;
			SELECT USER FROM MYSQL.
			InsuranceDetails;

SHOW DATABASE	Displays a list of databases created until present.	SELECT USER FROM MYSQL.
		DatabaseName;
			SELECT USER FROM MYSQL.
			InsuranceDetails;

SHOW TABLE	Displays a list of tables created in the database until present.	SHOW DATABASES;	SHOW DATABASES;

SHOW COLUMNS	Displays all the columns present in the database.	SHOW COLUMNS FROM TableName FROM DatabaseName;
		Or:
		SHOW COLUMNS FROM DatabaseName.
		TableName;
			SHOW COLUMNS FROM CarInsurance FROM InsuranceDetails;
			Or:
			SHOW COLUMNS FROM InsuranceDetails.
			CarInsurance;

USE DATABASE	Tells the system which database to be chosen to perform actions.	USE DatabaseName;	USE InsuranceDetails;

TABLE CONTINUES ON THE NEXT PAGE

COMMAND	DESCRIPTION	SYNTAX	EXAMPLE

`ALTER TABLE`	Modifies tables by adding, deleting, or modifying data in columns.	`ALTER TABLE TableName`
		`ADD Column_1 datatype;`
			`ALTER TABLE CarInsurance`
			`ADD PolicyCreatedDate datetime;`

`DESCRIBE TABLE`	Displays the structure of the table consisting of column names, data types, keys and default values.	`DESCRIBE TableName;`	`DESCRIBE CarInsurance;`
`TRUNCATE TABLE`	Deletes only the data present in tables without deleting the table.	`TRUNCATE TABLE TableName;`	`TRUNCATE TABLE CarInsurance;`
`RENAME DATABASE`	Renames a database.	`RENAME TABLE Database1 TO Database2;`	`RENAME TABLE InsuranceDetails TO Insurances;`
`RENAME TABLE`	Renames a table.	`RENAME TABLE Table1 TO Table2;`	`RENAME TABLE CarInsurance TO CarInsuranceDetails;`

`CHANGE COLUMNS`	Changes column names of a table.	`ALTER TABLE TableName`
		`CHANGE COLUMN OldColumn NewColumn DataType;`
		Or:
		To change multiple column names with the `CHANGE` query:
		`ALTER TABLE TableName (`
		`CHANGE COLUMN OldColumn NewColumn DataType,`
		`CHANGE COLUMN OldColumn NewColumn DataType);`
			`ALTER TABLE CarInsurance`
			`CHANGE COLUMN PolicyName PName VARCHAR(255);`
			Or:
			To change multiple column names with the `CHANGE` query:
			`ALTER TABLE CarInsurance(`
			`CHANGE COLUMN PolicyName PName VARCHAR(255),`
			`CHANGE COLUMN EffectiveDate EDate DateTime);`

`RENAME COLUMNS`	Renames columns present in a table.	`ALTER TABLE TableName`
		`RENAME COLUMN OldColumn TO NewColumn;`
		Or:
		To change multiple column names with `RENAME` query:
		`ALTER TABLE TableName (`
		`RENAME COLUMN OldColumn TO NewColumn,`
		`RENAME COLUMN OldColumn TO NewColumn);`
			`ALTER TABLE CarInsurance`
			`RENAME COLUMN PolicyName TO PName;`
			Or:
			`ALTER TABLE CarInsurance(`
			`RENAME COLUMN PolicyName TO PName,`
			`RENAME COLUMN EffectiveDate TO EDate);`

`INSERT INTO`	Inserts new records into a table.	`INSERT INTO TableName (Column_1, Column_2, ...)`
		`VALUES (Value_1, Value_2,...);`
		Or:
		`INSERT INTO TableName`
		`VALUES (Value_1, Value_2);`
			`INSERT INTO TableName (PolicyID, PolicyName, EffectiveDate, ExpiryDate, PaymentOption, Amount, Status)`
			`VALUES ('567', 'SampleInsurance', '03-01-2022', '03-01-2037', 'Debit Card', '12000', 'TRUE');`

TABLE CONTINUES ON THE NEXT PAGE

COMMAND	DESCRIPTION	SYNTAX	EXAMPLE

			Or:
			`INSERT INTO CarInsurance`
			`VALUES ('567', 'SampleInsurance',`
			`'03-01-2022', '03-01-2037', 'Debit`
			`Card', '12000', 'TRUE');`

`UPDATE TABLE`	Modifies the existing data items in a table.	`UPDATE TableName`
		`SET Column1 = Value1, Column2 =`
		`Value2, ...`
		`WHERE condition;`
			`UPDATE CarInsurance`
			`SET PaymentOption = 'Credit Card'`
			`WHERE PolicyID = '567';`

`DELETE COLUMNS`	Deletes column in table.	`ALTER TABLE TableName`
		`    DROP COLUMN Column_1;`
		Or:
		To drop multiple columns:
		`ALTER TABLE TableName   (`
		`    DROP COLUMN Column_2,`
		`    DROP COLUMN Column_3);`
			`ALTER TABLE CarInsurance`
			`    DROP COLUMN PaymentOption;`
			Or:
			To drop multiple columns:
			`ALTER TABLE TableName   (`
			`    DROP COLUMN PolicyName,`
			`    DROP COLUMN ExpiryDate);`

`ADD COLUMN`	Adds a new column in the table.	`ALTER TABLE TableName`
		`    ADD COLUMN NewColumn_1 DataType;`
		Or:
		To add after a particular column:
		`ALTER TABLE TableName`
		`    ADD COLUMN NewColumn_1 DataType`
		`AFTER Column_2;`
		Or:
		To add multiple columns:
		`ALTER TABLE TableName   (`
		`    ADD COLUMN NewColumn_1 DataType`
		`AFTER Column_2,`
		`    ADD COLUMN NewColumn_1 DataType`
		`AFTER Column_2 );`
			`ALTER TABLE CarInsurance`
			`    ADD COLUMN DOB datetime;`
			Or:
			To add after a particular column:
			`ALTER TABLE CarInsurance`
			`    ADD COLUMN DOB datetime AFTER`
			`PaymentOption;`
			Or:
			To add multiple columns:
			`ALTER TABLE CarInsurance (`
			`    ADD COLUMN DOB datetime AFTER`
			`PaymentOption,`
			`    ADD COLUMN Description`
			`varchar(255));`

`SELECT`	Selects data values from a database and stores the data returned as output in the result set.	`SELECT Column_1, Column_2, ...`
		`FROM TableName;`
			`SELECT PolicyID, PolicyName FROM`
			`CarInsurance;`

`SELECT FROM`	Retrieves data values from a table.	`SELECT Column_1, Column_2, ...`
		`FROM TableName;`
			`SELECT PolicyID, PolicyName FROM`
			`CarInsurance;`

`SELECT DISTINCT`	Returns only distinct values from the database.	`SELECT DISTINCT Column_1, Column_2,`
		`...`
		`FROM TableName;`
			`SELECT DISTINCT EffectiveDate FROM`
			`CarInsurance;`

`SELECT WHERE`	Filters data based on conditions.	`SELECT Column_1, Column_2, ...`
		`FROM TableName WHERE Conditions;`
			`SELECT PolicyName, EffectiveDate`
			`FROM CarInsurance WHERE PolicyID =`
			`'567';`

CONCLUSION

Database administrators, data analysts, data scientists, and many other professionals use MySQL daily to generate meaningful insights and make informed decisions. We hope you found this Refcard informative and now understand the essentials of MySQL. Practice these commands and take your MySQL knowledge to the next level.

WRITTEN BY SAHITI KAPPAGANTULA,
PRODUCT ASSOCIATE, OORWIN

Having 4+ years of experience, I am a product enthusiast with immense interest in technology and marketing. I have worked on technologies such as MySQL, R, microservices, DevOps, and RPA. I enjoy building game-changing products from scratch and delivering them to consumers by implementing product and growth strategies.

600 Park Offices Drive, Suite 300
Research Triangle Park, NC 27709
888.678.0399 | 919.678.0300

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community. We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.