1. Introduction to SQL(Definition, parts, General structure of Query)

SQL is a standard language for accessing and manipulating databases.

- SQL stands for Structured Query Language.
- SQL lets you access and manipulate databases.
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.

RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

General structure of Query

- CREATE DATABASE ***databasename***;

  Eg:CREATE DATABASE ***test***;

- CREATE TABLE ***table_name***(column1 datatype,column2 datatype,column3 datatype,....column n datatype;);

  Eg: Table with constraint:- Persons(PersonID, LastName, FirstName, Address, City)

- CREATE TABLE ***table_name***(column1 datatype ***constraint***, column2 datatype ***constraint***, column3 datatype ***constraint***,.....);

  Eg: CREATE TABLE STUD (SID INT PRIMARY KEY NOT NULL, SNAME VARCHAR (10) NOT NULL, MARK INT);

  The following constraints are commonly used in SQL:

  NOT NULL - Ensures that a column cannot have a NULL value

  UNIQUE - Ensures that all values in a column are different

  PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - Prevents actions that would destroy links between tables

CHECK - Ensures that the values in a column satisfies a specific condition

DEFAULT - Sets a default value for a column if no value is specified

CREATE INDEX - Used to create and retrieve data from the database very quickly

Structured Query Language (SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also, we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

DDL – Data Definition Language

DML – Data Manipulation Language

DCL – Data Control Language

TCL –Transaction Control Language

List of DDL commands:

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

DROP: This command is used to delete objects from the database.

ALTER: This is used to alter the structure of the database.

TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.

COMMENT: This is used to add comments to the data dictionary.

RENAME: This is used to rename an object existing in the database.

List of DML commands:

INSERT : It is used to insert data into a table.

SELECT: It is used to retrieve data from the database.

UPDATE: It is used to update existing data within a table.

DELETE : It is used to delete records from a database table.

LOCK: Table control concurrency.

CALL: Call a PL/SQL or JAVA subprogram.

EXPLAIN PLAN**:** It describes the access path to data.

List of DCL commands:

GRANT: This command gives users access privileges to the database.

REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

List of TCL commands:

COMMIT: Commits a Transaction.

ROLLBACK: Rollbacks a transaction in case of any error occurs.

SAVEPOINT: Sets a savepoint within a transaction.

SET TRANSACTION: Specify characteristics for the transaction.

INSERT STATEMENT

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)VALUES (*value1*, *value2*, *value3*, ...);

OR

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO *table_name VALUES* (*value1*, *value2*, *value3*, ...);

Eg: INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

INSERT INTO Customers (CustomerName, City, Country)VALUES('Cardinal', 'Stavanger', 'Norway');

SELECT STATEMENT

The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

Syntax

SELECT *column1*, *column2, ...FROM table_name*;

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT * FROM *table_name*;

Eg: SELECT CustomerName, City FROM Customers;

OR

The following SQL statement selects all the columns from the "Customers" table:

Eg: SELECT * FROM Customers;

ALTER STATEMENT

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

- ALTER TABLE - ADD Column
  ALTER TABLE ***table_name*** ADD ***column_name datatype***;
- ALTER TABLE - DROP Column

  ALTER TABLE ***table_name*** DROP COLUMN ***column_name***;
- ALTER TABLE - MODIFY/ALTER Column

  ALTER TABLE ***table_name*** ALTER/MODIFY COLUMN ***column_namedatatype***;

  For ORACLE 10 G

  ALTER TABLE ***table_name MODIFY column_name datatype***;

  Eg: ALTER TABLE Persons ALTER COLUMN DateOfBirth year;

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a column it will allow only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

RENAME STATEMENT

ALTER TABLE table_name RENAME TO new_table_name;

DROP STATEMENT

DROP command completely removes a table from the database. This command will also destroy the table structure and the data stored in it. Following is its syntax,

DROP TABLE table_name;
Eg: DROP TABLE student;

TRUNCATE STATEMENT

The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.

You can also use DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

Syntax:

The basic syntax of a TRUNCATE TABLE command is as follows.

TRUNCATE TABLE table_name;

UPDATE STATEMENT

The UPDATE statement is used to modify the existing records in a table.

Syntax

UPDATE *table_name SET column1 = value1*, *column2 = value2*, ...WHERE *condition*;

Eg: UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;

DELETE STATEMENT

The DELETE statement is used to delete existing records in a table.

Syntax

DELETE FROM *table_name* WHERE *condition*;

Eg: DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

STRING FUNCTIONS

These functions are used to perform an operation on input string and return an output string.

Following are the string functions defined in SQL:

- **ASCII():** This function is used to find the ASCII value of a character.

  Syntax: SELECT ascii('t');

  Output: 116

- **CHAR_LENGTH():** Doesn't work for SQL Server. Use LEN() for SQL Server. This function is used to find the length of a word.

  Syntax: SELECT char_length('Hello!');

  Output: 6

- **CHARACTER_LENGTH():** Doesn't work for SQL Server. Use LEN() for SQL Server. This function is used to find the length of a line.

  Syntax: SELECT CHARACTER_LENGTH('geeks for geeks');

  Output: 15

- **CONCAT():** This function is used to add two words or strings.

  Syntax: SELECT 'Geeks' || ' ' || 'forGeeks' FROM dual;

  Output: 'GeeksforGeeks'

- **CONCAT_WS():** This function is used to add two words or strings with a symbol as concatenating symbol.

Syntax: SELECT CONCAT_WS('_', 'geeks', 'for', 'geeks');

Output: geeks_for_geeks

- **FIND_IN_SET():** This function is used to find a symbol from a set of symbols.

  Syntax: SELECT FIND_IN_SET('b', 'a, b, c, d, e, f');

  Output: 2

- **FORMAT():** This function is used to display a number in the given format.

  Syntax: Format("0.981", "Percent");

  Output: '98.10%'

- **INSERT():** This function is used to insert the data into a database.

  Syntax: INSERT INTO database (geek_id, geek_name) VALUES (5000, 'abc');

  Output: successfully updated

- **INSTR():** This function is used to find the occurrence of an alphabet.

  Syntax: INSTR('geeks for geeks', 'e');

  Output: 2 (the first occurrence of 'e')

  Syntax: INSTR('geeks for geeks', 'e', 1, 2);

  Output: 3 (the second occurrence of 'e')

- **LCASE():** This function is used to convert the given string into lower case.

  Syntax: LCASE ("GeeksFor Geeks To Learn");

  Output: geeksforgeeks to learn

- **LEFT():** This function is used to SELECT a sub string from the left of given size or characters.

  Syntax: SELECT LEFT('geeksforgeeks.org', 5);

  Output: geeks

- **LENGTH():** This function is used to find the length of a word.

  Syntax: LENGTH('GeeksForGeeks');

  Output: 13

- **LOCATE():** This function is used to find the nth position of the given word in a string.

  Syntax: SELECT LOCATE('for', 'geeksforgeeks', 1);

  Output: 6

- **LOWER():** This function is used to convert the upper case string into lower case.

  Syntax: SELECT LOWER('GEEKSFORGEEKS.ORG');

  Output: geeksforgeeks.org

- **LPAD():** This function is used to make the given string of the given size by adding the given symbol.

  Syntax: LPAD('geeks', 8, '0');

  Output: 000geeks

- **Various Aggregate Functions**

  1) Count()

  2) Sum()

  3) Avg()

  4) Min()

  5) Max()

2. **Using the tables, employee(emp_id, empfname, emplname, dept, pjt, address, DOB, gender) empdesig(emp_id, emp_position, dateofjoin, salary). Write queries to:**

   a) **Create tables and insert the values.**

Create table EMPLOYEE(EMPLOYEE_ID varchar(5) NOT NULL PRIMARY KEY, EMP_FNAME varchar (20) NOT NULL, EMP_LNAME varchar(25) , DEPARTMENT varchar (30) ,PJT varchar(30),ADDRESS varchar(30),DOB date, GENDER varchar (6));

Insert into EMPLOYEE VALUES('EMP01','ATHIRA','PRADEEP','TECHANICAL','PROJECTA','TRIVANDRUM','12-12-1998','FEMALE');

Insert into EMPLOYEE VALUES('EMP02','RASIYA','ANASI','SALES','PROJECTB','KOCHI','01-02-1988','FEMALE');

Insert into EMPLOYEE VALUES('EMP03','RAM','ANAND','MARKETING','PROJECTC','KOCHI','11-02-1993','MALE');

Insert into EMPLOYEE VALUES('EMP04','VYSHAK','VIJAY','HUMAN RESOURCES','PROJECTD','TRIVANDRUM','10-05-1995','MALE');

Insert into EMPLOYEE VALUES('EMP05','ASHWIN','KUMAR','HUMAN RESOURCES','PROJECTE','TRIVANDRUM','20-05-1994','MALE');

Insert into EMPLOYEE VALUES('EMP06','SHIVANGI','AKASH','TECHANICAL','PROJECTA','TRIVANDRUM','22-08-1997','FEMALE');

SELECT * FROM EMPLOYEE;

	employee_id [PK] character vary	emp_fname character varying (	emp_lname character varying	department character varying (30)	pjt character varying (3(	address character varying (3(	dob date	gender character varying
1	EMP01	ATHIRA	PRADEEP	TECHANICAL	PROJECTA	TRIVANDRUM	1998-12-12	FEMALE
2	EMP02	RASIYA	ANASI	SALES	PROJECTB	KOCHI	1988-02-01	FEMALE
3	EMP03	RAM	ANAND	MARKETING	PROJECTC	KOCHI	1993-02-11	MALE
4	EMP06	SHIVANGI	AKASH	TECHANICAL	PROJECTA	TRIVANDRUM	1997-08-22	FEMALE
5	EMP07	MINI	THOMAS	SALES	PROJECTB	KOCHI	1995-02-12	FEMALE
6	EMP04	VYSHAK	VIJAY	HUMAN RESOURCES	PROJECTD	TRIVANDRUM	1995-05-10	MALE
7	EMP05	ASHWIN	KUMAR	HUMAN RESOURCES	PROJECTE	TRIVANDRUM	1994-05-20	MALE

Creating table EMPDESIG

Create table EMPDESIG(EMPLOYEE_ID varchar(5) references EMPLOYEE(EMPLOYEE_ID) PRIMARY KEY,EMPLOYEE_POSITION varchar(30),DOJ date,SALARY int);

Insert into EMPDESIG(‘EMP01’,’PROGRAMMER’,’05-05-2020’,30000);

Insert into EMPDESIG(‘EMP02’,’SALESMAN’,’10-03-2018’,25000);

Insert into EMPDESIG(‘EMP03’,’MANAGER’,’19-03-2017’,50000);

Insert into EMPDESIG(‘EMP04’,’MANAGER’,’11-06-2012’,60000);

Insert into EMPDESIG(‘EMP05’,’MANAGER’,’16-11-2019’,55000);

Insert into EMPDESIG(‘EMP06’,’PROGRAMMER,’22-05-2012’,55000);

Insert into EMPDESIG(‘EMP07’,’CLERK’,’20-03-2017’,35000);

SELECT * FROM EMPDESIG;

	employee_id [PK] character varying (5)	employee_position character varying (30)	doj date	salary integer
1	EMP01	PROGRAMMER	2020-05-05	30000
2	EMP02	SALESMAN	2018-03-10	25000
3	EMP03	MANAGER	2017-03-19	50000
4	EMP04	MANAGER	2012-06-11	60000
5	EMP05	MANAGER	2019-11-16	55000
6	EMP06	PROGRAMMER	2015-05-22	55000
7	EMP07	CLERK	2017-03-20	35000

b) Find the salary of all clerks and managers?

Select SALARY from EMPDESIG where EMPLOYEE_POSITION='CLERK' OR EMPLOYEE_POSITION='MANAGER';

	salary integer
1	50000
2	60000
3	55000
4	35000

c) Fetch the no of employees working in the "HR" department?

Select count(*) from EMPLOYEE where DEPARTMENT='HUMAN RESOURCES';

	count bigint
1	2

d) Fetch the first four characters of emplname from the employee table?

SELECT SUBSTRING(EMP_LNAME,1,4) FROM employee ;

	substring text
1	PRAD
2	ANAS
3	ANAN
4	VIJA
5	KUMA
6	AKAS
7	THOM

e) Find all employee details whose salary is between 10000 and 40000?

SELECT * FROM EMPDESIG WHERE salary >=10000 AND salary <=40000;

	employee_id [PK] character varying (5)	employee_position character varying (30)	doj date	salary integer
1	EMP01	PROGRAMMER	2020-05...	30000
2	EMP02	SALESMAN	2018-03...	25000
3	EMP07	CLERK	2017-03...	35000

f) Create a new table which consists of data and structure copied from employee table?

CREATE TABLE newemployee AS SELECT * FROM employee;

Select * from newemployee;

employee_id character varying (5)	emp_fname character varying (20)	emp_lname character varying (25)	department character varying (30)	pjt character varying (30)	address character varying (30)	dob date	gender character varying (6)
EMP01	ATHIRA	PRADEEP	TECHANICAL	PROJECTA	TRIVANDRUM	1998-12...	FEMALE
EMP02	RASIYA	ANASI	SALES	PROJECTB	KOCHI	1988-02...	FEMALE
EMP03	RAM	ANAND	MARKETING	PROJECTC	KOCHI	1993-02...	MALE
EMP04	VYSHAK	VIJAY	HUMAN RESOURCES	PROJECTD	TRIVANDRUM	1995-05...	MALE
EMP05	ASHWIN	KUMAR	HUMAN RESOURCES	PROJECTE	TRIVANDRUM	1994-05...	MALE
EMP06	SHIVANGI	AKASH	TECHANICAL	PROJECTA	TRIVANDRUM	1997-08...	FEMALE
EMP07	MINI	THOMAS	SALES	PROJECTB	KOCHI	1995-02...	FEMALE

g) To find details of employees whose emplname ends with ‘A’ and contains five alphabets?

SELECT * FROM employee WHERE EMP_LNAME LIKE 'A%' and length(EMP_LNAME)=5;

employee_id [PK] character varying (5)	emp_fname character varying (20)	emp_lname character varying (25)	department character varying (30)	pjt character varying (30)	address character varying (30)	dob date	gender character varying
EMP02	RASIYA	ANASI	SALES	PROJECTB	KOCHI	1988-02...	FEMALE
EMP03	RAM	ANAND	MARKETING	PROJECTC	KOCHI	1993-02...	MALE
EMP06	SHIVANGI	AKASH	TECHANICAL	PROJECTA	TRIVANDRUM	1997-08...	FEMALE

h) To add email validation to your database?

ALTER TABLE employeeADD Email varchar(255);

SELECT emailFROM employeeWHERE NOT REGEXP_LIKE(email,'[A-Z0-9.-%+-]+@[A-Z0-9.-]+\.[A-Z]{2-4}','$');

Data Output Explain Messages Notifications

```
ALTER TABLE

Query returned successfully in 158 msec.
```

i) To find the second highest salary?

SELECT * FROM EMPDESIG A WHERE 2= (SELECT COUNT(DISTINCT salary) FROM EMPDESIG B WHERE A.salary<=B.salary);

	employee_id [PK] character varying (5)	employee_position character varying (30)	doj date	salary integer
1	EMP05	MANAGER	2019-11...	55000
2	EMP06	PROGRAMMER	2015-05...	55000

3. Introduction to plsql(definition,basic structure of plsql block)

PL/SQL Introduction

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements. All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Disadvantages of SQL:

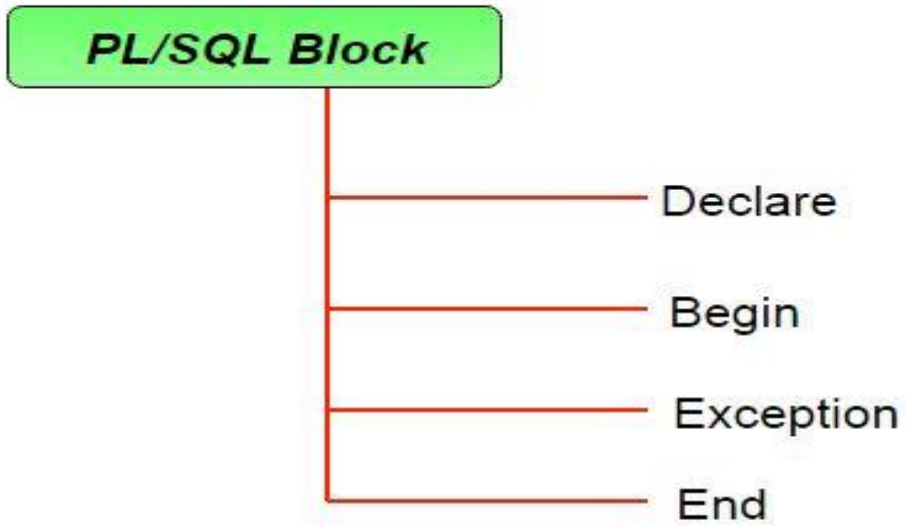
- SQL doesn't provide the programmers with a technique of condition checking, looping and branching.
- SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.
- SQL has no facility of error checking during manipulation of data.

Features of PL/SQL:

1. PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.
2. PL/SQL can execute a number of queries in one block using single command.
3. One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.
4. PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
5. Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.
6. PL/SQL Offers extensive error checking.

Structure of PL/SQL Block:

PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL. The basic unit in PL/SQL is a block. All PL/SQL programs are made up of blocks, which can be nested within each other.



DECLARE:

- Declare section starts with **DECLARE** keyword in which variables, constants, records as cursors can be declared which stores data temporarily. It basically consists definition of PL/SQL identifiers. This part of the code is optional.

BEGIN:

- Execution section starts with **BEGIN** and ends with **END** keyword. This is a mandatory section and here the program logic is written to perform any task like loops and conditional statements. It supports all DML commands, DDL commands and SQL*PLUS built-in functions as well.

EXCEPTIONS:

- Exception section starts with **EXCEPTION** keyword. This section is optional which contains statements that are executed when a run-time error occurs. Any exceptions can be handled in this section.

END: end of the statement.

PL/SQL identifiers

There are several PL/SQL identifiers such as variables, constants, procedures, cursors, triggers etc.

Variables:

Like several other programming languages, variables in PL/SQL must be declared prior to its use. They should have a valid name and data type as well.

Syntax for declaration of variables:

variable_name datatype [NOT NULL: = value];

Using Comments:

Like in many other programming languages, in PL/SQL also, comments can be put within the code which has no effect in the code. There are two syntaxes to create comments in PL/SQL:

- **Single Line Comment:** To create a single line comment, the symbol – – is used.
- **Multi Line Comment:** To create comments that span over several lines, the symbol /* and */ is used.

Displaying Output:

The outputs are displayed by using DBMS_OUTPUT which is a built-in package that enables the user to display output, debugging information, and send messages from PL/SQL blocks, subprograms, packages, and triggers.

4.Write a plsql program

a) To display the message welcome to plsql".

```
CREATE OR REPLACE PROCEDURE greetings AS BEGIN
dbms_output.put_line('Welcome to plsql');
 END;
/
```

OUTPUT

Welcome to plsql

PL/SQL procedure successfully completed.

b) To check whether a given character is a letter or a digit.

```
DECLARE
get_ctr CHAR(1) := '&input_a_character';
BEGIN
IF ( get_ctr >= 'A'
AND get_ctr <= 'Z' )
OR ( get_ctr >= 'a'
AND get_ctr <= 'z' ) THEN
dbms_output.Put_line ('The given character is a letter');
ELSE
dbms_output.Put_line ('The given character is not a letter');
IF get_ctr BETWEEN '0' AND '9' THEN
dbms_output.Put_line ('The given character is a number');
ELSE
dbms_output.Put_line ('The given character is not a number');
END IF;
END IF;
END;
/
```

OUTPUT

```
SQL> /
Enter value for input_a_character: m
The given character is a letter
PL/SQL procedure successfully completed.
```

5. **Write a plsql program to check whether a given number is palindrome or not.**

```
declare
n number;
m number;
rev number:=0;
r number;
begin
n:=12321;
m:=n;
while n>0
loop
r:=mod(n,10);
rev:=(rev*10)+r;
n:=trunc(n/10);
end loop;
if m=rev
then
dbms_output.put_line('number is palindrome');
else
dbms_output.put_line('number is not palindrome');
end if;
end;
/
```

OUTPUT

```
number is palindrome
PL/SQL procedure successfully completed.
```

6. **Write a PL/SQL block to calculate the incentive (20% of salary) of an employee whose emp_id is 10 using the above empdesig table.**

```
DECLARE
incentive NUMBER;
BEGIN
SELECT salary * 0.20 INTO incentive FROM EMPDESIG WHERE employee_id = EMP01;
DBMS_OUTPUT_LINE('Incentive ='|| TO_CHAR(incentive));\
END;
/
```

OUTPUT

```
Incentive =6000
PL/SQL procedure successfully completed.
```

7.Introduction to NoSQL (Definition,advantages,structure).

Introduction to NoSQL

A **NoSQL** originally referring to non SQL or non-relational is a database that provides a mechanism for storage and retrieval of data. This data is modelled in means other than the tabular relations used in relational databases. Such databases came into existence in the late 1960s, but did not obtain the NoSQL moniker until a surge of popularity in the early twenty-first century. NoSQL databases are used in real-time web applications and big data and their use are increasing over time. NoSQL systems are also sometimes called Not only SQL to emphasize the fact that they may support SQL-like query languages.

A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines and finer control over availability. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it should solve. Data structures used by NoSQL databases are sometimes also viewed as more flexible than relational database tables.

Many NoSQL stores compromise consistency in favor of availability, speed and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, lack of standardized interfaces, and huge previous investments in existing relational databases. Most NoSQL stores lack true ACID(Atomicity, Consistency, Isolation, Durability) transactions but a few databases, such as MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made them central to their designs.

Most NoSQL databases offer a concept of eventual consistency in which database changes are propagated to all nodes so queries for data might not return updated data immediately or might result in reading data that is not accurate which is a problem known as stale reads. Also some NoSQL systems may exhibit lost writes and other forms of data loss. Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss. For distributed transaction processing across multiple databases, data consistency is an even bigger challenge. This is difficult for both NoSQL and relational databases. Even current relational databases do not allow referential integrity constraints to span databases. There are few systems that maintain both X/Open XA standards and ACID transactions for distributed transaction processing.

Advantages of NoSQL:

There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

High scalability –

NoSQL database use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra etc. NoSQL can handle huge amount of data because of scalability, as the data grows NoSQL scale itself to handle that data in efficient manner.

High availability –

Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

Disadvantages of NoSQL:

NoSQL has the following disadvantages.

Narrow focus –

NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.

Open-source –

NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words, two database systems are likely to be unequal.

Management challenge –

The purpose of big data tools is to make management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.

GUI is not available –

GUI mode tools to access the database is not flexibly available in the market.

Backup –

Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.

Large document size –

Some database systems like MongoDB and CouchDB store data in JSON format.

Which means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts, since they increase the document size.

Types of NoSQL database:

Types of NoSQL databases and the name of the databases system that falls in that category are:

MongoDB falls in the category of NoSQL document based database.

Key value store: Memcached, Redis, Coherence

Tabular: Hbase, Big Table, Accumulo

Document based: MongoDB, CouchDB, Cloudant

When should NoSQL be used:

- When huge amount of data needs to be stored and retrieved.
- The relationship between the data you store is not that important
- The data changing over time and is not structured.
- Support of Constraints and Joins is not required at database level
- The data is growing continuously and you need to scale the database regular to handle the data.

8.Compare relational and nonrelational databases.

RELATIONAL DATABSE	NON-RELATIONAL DATABSE
• A relational database is one that stores data in tables. • The relationship between tables and field types is called a schema. • Relational databases are also called SQL databases. • A relational database works by linking information from multiple tables through the use of "*keys*." • Popular relational databases are SQL Server, MySQL, PostgreSQL • They work with structured data. • There are limitless indexing capabilities, which results in faster query response times. • They are excellent at keeping data transactions secure. • They are table and row oriented. • SQL databases are best fit for heavy duty transactional type applications.	• A non-relational database is any database that does not use the tabular schema of rows and columns like in relational databases. • They provide schema-free or schema-on-read options. • Non-relational databases are also known as NoSQL databases which stands for "Not Only SQL." • They have the ability to store large amounts of data with little structure. • They are document oriented. • popular non-relational databases are MongoDB, Redis, Apache, Cassandra • They provide scalability and flexibility to meet changing business requirements. • They provide flexible data model with the ability to easily store and combine data of any structure without the need to modify a schema. • They are best for Rapid Application Development. NoSQL is the best selection for flexible data storage with little to no structure limitations. • They have the ability to capture all types of data "Big Data" including unstructured data.

9. Write steps to install and configure NoSQL databases.

MongoDB Community Edition on Windows

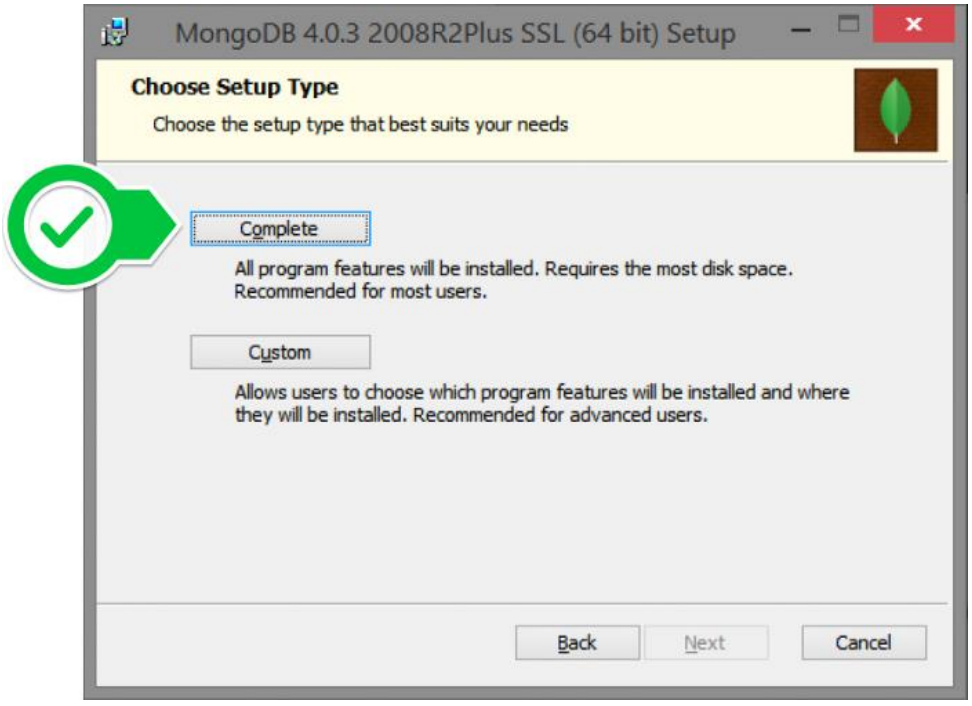
MongoDB, the most popular database for modern apps, and MongoDB Atlas, the global cloud database on AWS, Azure, and GCP. MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

Step1—Download the MongoDB MSI Installer Package

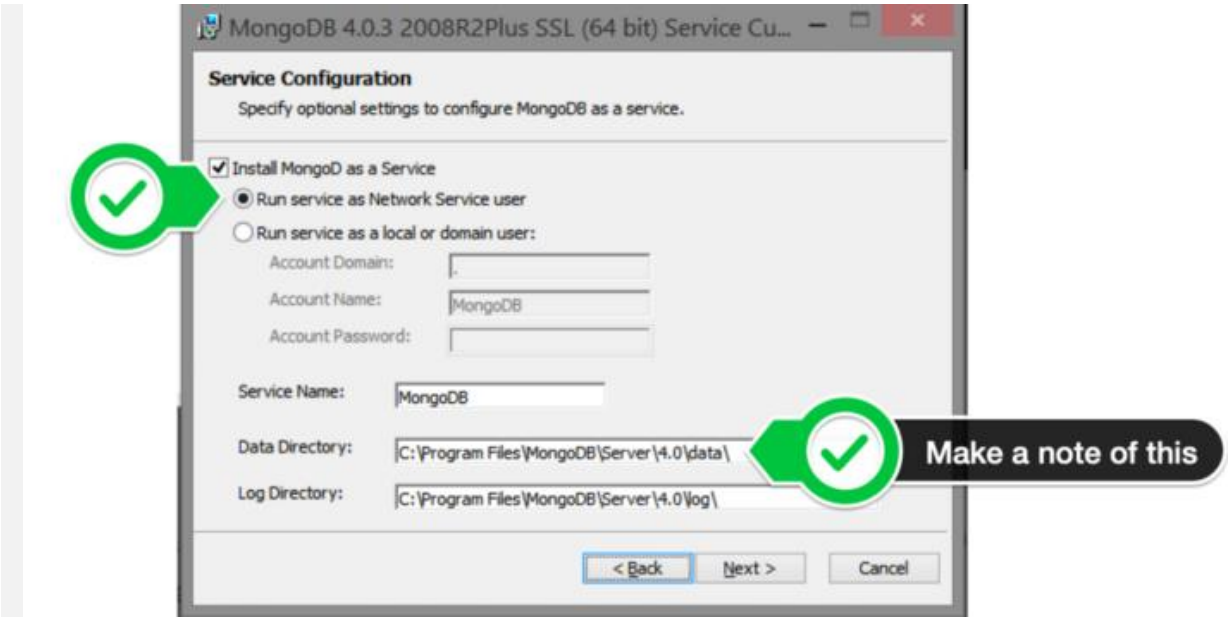
Make sure you select MSI as the package you want to download.

Step 2 — Install MongoDB with the Installation Wizard

A. Make sure you are logged in as a user with Admin privileges. Then navigate to your downloads folder and double click on the .msi package you just downloaded. This will launch the installation wizard.
B. Click Next to start installation.
C. Accept the licence agreement then click Next.
D. Select the Complete setup.



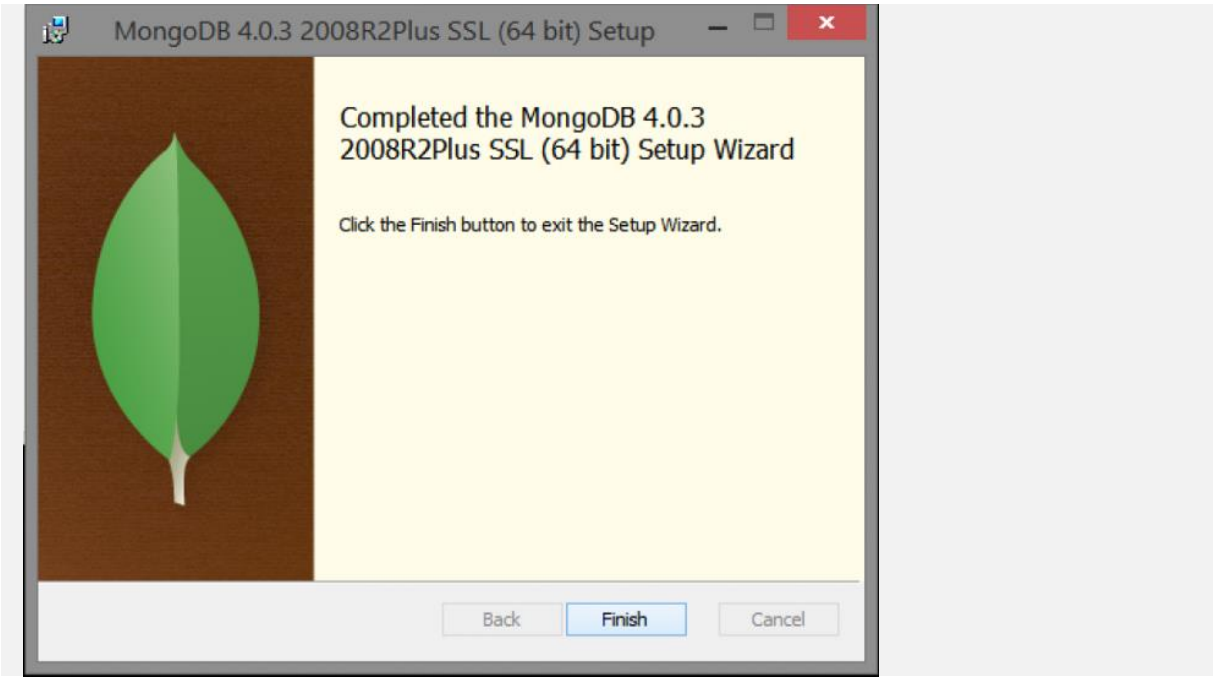
E. Select “Run service as Network Service user” and make a note of the data directory, we’ll need this later.



F. We won’t need Mongo Compass, so deselect it and click Next.

G. . Click Install to begin installation.

H. Hit Finish to complete installation.



Step 3— Create the Data Folders to Store our Databases

A. Navigate to the C Drive on your computer using Explorer and create a new folder called data here.

B. Inside the data folder you just created, create another folder called db.

Step 4— Setup Alias Shortcuts for Mongo and Mongod

Once installation is complete, we'll need to set up MongoDB on the local system.

A. Open up your Hyper terminal running Git Bash.

B. Change directory to your home directory with the following command:

```
cd~
```

C. Here, we're going to create a file called .bash_profile using the following command:

```
touch .bash_profile
```

D. Open the newly created .bash_profile with vim using the following command:

```
vim .bash_profile
```

E. In vim, hit the **I** key on the keyboard to enter insert mode.

F. In your explorer go to C → Program Files → MongoDB → Server

Now you should see the version of your MongoDB.

G. Paste in the following code into vim, make sure your replace the 4.0 with your version that you see in explorer

```
alias mongod="/c/Program\ files/MongoDB/Server/4.0/bin/mongod.exe"
alias mongo="/c/Program\ Files/MongoDB/Server/4.0/bin/mongo.exe"
```

F. Hit the Escape key on your keyboard to exit the insert mode. Then type

```
:wq!
```

to save and exit Vim.

Step 5 — Verify That Setup was Successful

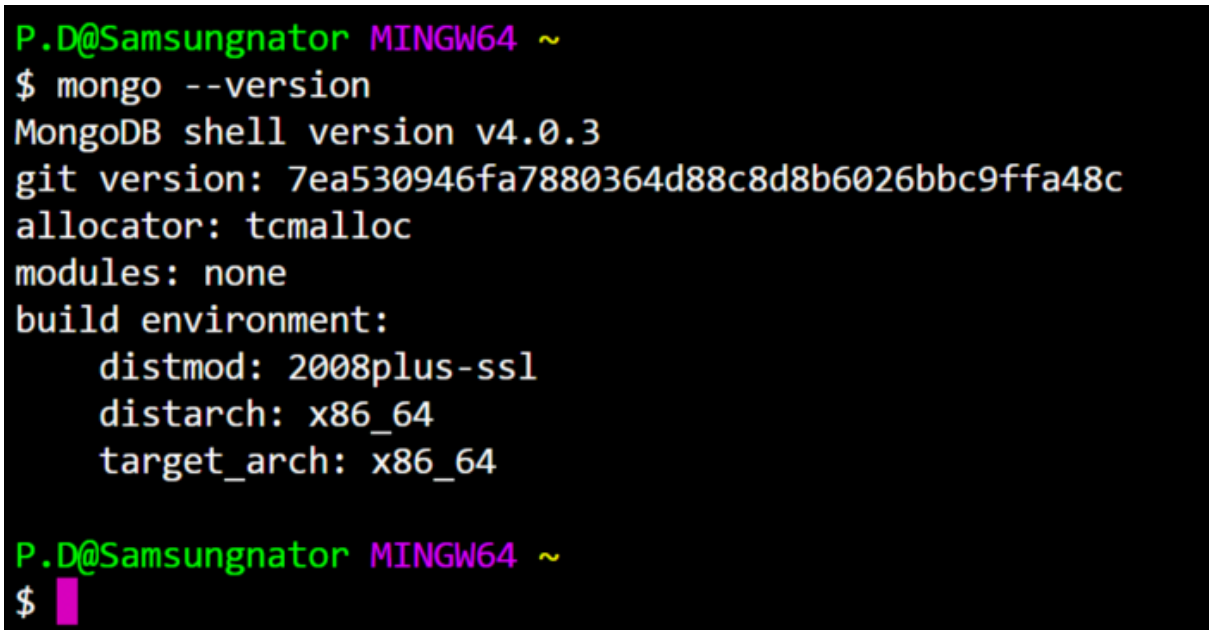
A. Close down the current Hyper terminal and quit the application.

B. Re-launch Hyper.

C. Type the following commands into the Hyper terminal:

```
mongo --version
```

Once you've hit enter, you should see something like this:



This means that you have successfully installed and setup MongoDB on your local system!