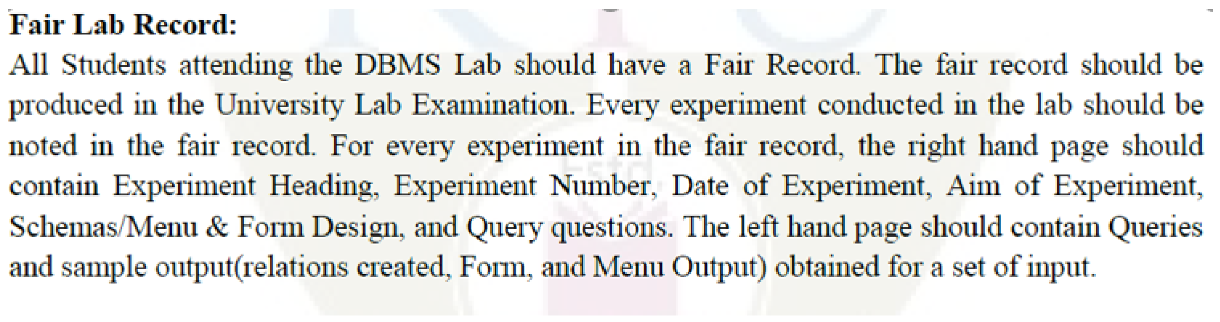
# FAIR RECORD FORMAT



**Fair Lab Record:**

All Students attending the DBMS Lab should have a Fair Record. The fair record should be produced in the University Lab Examination. Every experiment conducted in the lab should be noted in the fair record. For every experiment in the fair record, the right hand page should contain Experiment Heading, Experiment Number, Date of Experiment, Aim of Experiment, Schemas/Menu & Form Design, and Query questions. The left hand page should contain Queries and sample output(relations created, Form, and Menu Output) obtained for a set of input.

*******************************************************

Date: 10-12-2021 **(Batch1)**

14-12-2021 **(Batch 2)**

**Practice SQL commands for DDL and DML**

Experiment No.1

**Aim:**

- To familiarize DDL and DML commands
- To familiarize various string and numeric functions
- To familiarize aggregate functions

**Questions:**

**Write the complete question**

**Result: All the queries are successfully executed and output obtained.**

**Left hand side - write queries and draw the results**

*****************************************************

Date: 17-12-2021 **(Batch1)**

21-12-2021 **(Batch 2)**

**Practice SQL commands for DDL and DML**

Experiment No.2

**Aim:**

- To familiarize DDL commands, enforcing relationships
- To familiarize DML commands

**Questions:**

**Write the complete question**

**Result: All the queries are successfully executed and output obtained.**

**Left hand side - write queries and draw the results**

************************************************************

Date: 04-01-2022 **(Batch 1)**

07-01-2022 **(Batch 2)**

**Implementation of Aggregate functions, Built-in functions, Order by, Group by & Having clause**

Experiment No.3

**Aim:**

- To familiarize DDL and DML commands, setting check constraints, enforcing relationships
- To familiarize with numeric, date built-in functions.
- To familiarize various aggregate functions, group by, having & order by clause

**Questions:**

1) Create table department with dno as Primary Key starting with letter D, dname not null, manager name , count of employees should not be >30.
2) Create table employee with the following attributes, eno primary key and first letter should be 'e',ename not null,salary should not be zero, dno foreign key referring dno of department,mgrno ,doj,designation,address,city values should be Cochin,Mumbai,Chennai,Delhi,Bangalore,Kolkatta.
3) Insert values into both tables
4) Display the minimum length of employee name from employee table.
5) Find daily salary of the employees in the table emp.Assume 30 days are in a month. Round daily salary into 2 positions. Then name the column as "daily pay"
6) Select minimum salary of the employee with designation hod
7) Find the number of job titles in emp table.
8) Count the number of departments which has hod as designation.
9) Retrieve average and sum of salary from the table

10) Find the name of the employee who has the highest salary.
11) Find the total number of employees segregated on the basis of department.
12) Select sum of salary of employees under each manager.
13) Select employee name and their manager name from tables emp and dept.
14) display the details of employees in the order of date of joining.
15) Display the details of department in descending order of employee count.
16) List the minimum salary of various categories of employees in various departments having salary greater than 1000.
17) Get current system date and time
18) Display the last date of this month.

**Result: All the queries are successfully executed and output obtained.**

**Left hand side - write queries and draw the results**

## Queries and results

1) create table dept(dno varchar(20) primary key check(dno like "d%"),dname varchar(10) not null,mgrname varchar(10),countemp int check(countemp<30));
2) create table emp(eno varchar(10) primary key check(eno like "e%"), ename varchar(10) not null,salary int check(salary>0),dnum varchar(10) references dept(dno),mgrno int,doj date,des varchar(10),addr varchar(10),city varchar(10) check(city in ("=kochi","mumbai","chennai","delhi","banglore","kolkatta")));
3) insert into dept …
   insert into emp…

4) select min(length(ename)) "Minimum Length" from emp;
5) select ename,ROUND(salary/30,2) "Daily Pay" from emp;
6) select min(salary) from emp where des="hod";
7) select count(distinct(des)) from emp;
8) select count(dnum) from emp where des like "hod";
9) select avg(salary),sum(salary) from emp;
10) select ename from emp where salary=(select max(salary) from emp);
11) select count(eno) from emp group by dnum;

12) select sum(salary) from emp group by mgrno;
13) select eno,ename,salary,mgrname from emp,dept where dept.dno=emp.dnum;
14) select eno,ename,salary,dnum,doj from emp order by doj;
15) select * from dept order by countemp desc;
16) select dnum,min(salary) from emp group by dnum having min(salary)>1000;
17) select current_time,curtime(),current_time();
    select current_date,curdate(),current_date();
18) select last_day(curdate()) last_day ;

*******************************************************

Date: 14-01-2022 **(Batch 1 & 2)online lab**

## Implementation of Nested queries and joins

Experiment No.4

### Aim:

- To familiarize with nested queries, set operations and join queries

**Questions:**

1. Create the following tables

   a) book_details(ISBN, title, MRP, publisher_id, author)

   b) publisher (publisher_id, publisher_name, city, state, country)

2. Add primary key ISBN in book_details table and publisher_id in publisher respectively.

3. Add foreign key publisher_id in book_details

4. Populate the tables.

5. Display the details of all books and publisher_name.

6. Retrieve the details of all publishers where publisher_name Starts with letter M.

7. Display all book titles and MRP whose MRP is greater than minimum MRP and publisher name starts with M.

8. Issue a query to find all the book titles whose publisher name is Wiley.

9. Perform inner join, left join, right join and full join on the tables.(instead of full join we can use union all in mysql)

10. List the name of books, price and city of publisher of all books.

11. List the details of books and their corresponding publisher details.

12. List all publishers, details of books published by each publisher.(same as right outer join query)

13. List the name of publishers which have got entry either in book_details or publisher table.

**Result: All the queries are successfully executed and output obtained.**

**Left hand side - write queries and draw the results**

## Queries and results

1. create table book_details(ISBN varchar(20),title varchar(20), MRP decimal,publisher_id varchar(20), author varchar(20));

   create table publisher(publisher_id varchar(20),publisher_name varchar(20),city varchar(20), state varchar(20),country varchar(20));

2. alter table book_details add primary key(ISBN);

   alter table publisher add primary key(publisher_id);

3. alter table book_details add foreign key(publisher_id) references publisher(publisher_id);

4. insert into publisher values(…);

   insert into book_details values(…);

5. select ISBN,title,MRP,publisher_name,author from publisher,book_details where publisher.publisher_id=book_details.publisher_id;

6. select * from publisher where publisher_name like 'm%';

7. select ISBN,title, MRP,author from book_details where MRP > (select min(MRP) from book_details) and title like 'm%';

8. select title from book_details, publisher where
   book_details.publisher_id=publisher.publisher_id and
   publisher_name='Wiley;

9. select * from book_details b inner join publisher p on
   b.publisher_id=p.publisher_id;
   **same as**
   select * from book_details b, publisher p where b.publisher_id =
   p.publisher_id;
   **same as**
   select * from book_details  inner join publisher
   using(publisher_id);

10.
   select * from book_details b left join publisher p on
   b.publisher_id=p.publisher_id;

   select * from book_details b right join publisher p on
   b.publisher_id=p.publisher_id;

   select * from book_details b left join publisher p on
   b.publisher_id=p.publisher_id UNION ALL select * from
   book_details b right join publisher p on
   b.publisher_id=p.publisher_id;

11. select title,MRP,city from book_details b,publisher p where
   b.publisher_id=p.publisher_id;

12. select * from book_details b,publisher p where
   b.publisher_id=p.publisher_id;

13. select * from publisher p right join book_details b on
   b.publisher_id=p.publisher_id;

14. select distinct publisher_name from publisher p, book_details b;

Date: 14-01-2022 **(Batch 1&2) online lab**

**Practice of SQL commands for creation of views**

Experiment No.5

**Aim:**

- To familiarize with creation of views

**Questions:**

1. Create table employee with the following fields:
   name , salary , dept_no , designation, city
2. Create a view of the table employee.
3. Apply a select query on the created view.
4. Populate emp_view with values.
5. Modify view to set dno of John as d7;
6. Define another view v2 that contains employee id,ename,salary,dept id,and dept name.(Create appropriate tables and combine tables using left join).
7. Display the values in v2

Queries

1. create table employee(…);
2. create view emp_view as select ename,salary,dno from employee;
3. select * from emp_view where dno="d4";
4. insert into emp_view values(…);
5. update emp_view set dno="d7" where ename="John";
6. create view v2 as select e.ename, e.salary, e.city, d.dno, d.dname from employee_view e left outer join dept d on d.dno=e.dno;
7. select * from v2

**Error**

```
mysql> update v2 set city=mvpa where dno="d6";
```

```
ERROR 1288 (HY000): The target table v2 of the UPDATE is not updatable
*Cause:    An attempt was made to insert or update columns of a join view which
           map to a non-key-preserved table.
*Action:   Modify the underlying base tables directly.
```

Date: 25-01-2022 **(Batch 1)**

01-02-2022 **(Batch 2)**

## Practice of TCL commands

Experiment No.6

**Aim:**

- To familiarize with TCL Commands

**Questions:**

1. Create the following tables

a) book_details(ISBN primary key, title, MRP, publisher_id, author)

b) publisher (publisher_id primary key, publisher_name, city, state, country)

2. Add foreign key publisher_id in book_details

3. Populate the tables.

4. Use insert, update and delete statements (transactions) and practice commit, rollback and savepoint commands.

Date: 4-02-2022 **(Batch 1-Roll no.31 to 61)**

08-02-2022 **(Batch 2- Roll no.1 to 30)**

**Practice of DCL commands**

Experiment No.7

**Aim:**

- To familiarize with DCL Commands

**Questions:**

1) Create a new user – your name and password from the super user.
2) Grant privileges to the new user for a table in the database
3) Revoke privileges from the user
4) Practice Grant and Revoke commands on various objects of the database.

Queries (Tutorial : https://dev.mysql.com/doc/refman/5.7/en/grant.html)

**GRANT**

```
Login to MySQL with the super user (root)
mysql -u root -p

mysql> CREATE USER 'newuser'@localhost IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.34 sec)

mysql> show databases;

mysql> GRANT ALL PRIVILEGES ON yourdbname.yourtablename  TO
'newuser'@'localhost';

SHOW GRANTS FOR 'newuser'@'localhost';

mysql> exit;
```

---

```
Login to MySQL with the new user

mysql -u newuser -p
Enter password: ****

mysql> use yourdbname;

mysql> show tables;
```

DDL and DML commands on the table selected.
mysql> exit;

---

Login to MySQL with the super user (root)

```
mysql -u root -p
```

```
CREATE USER 'user1'@localhost IDENTIFIED BY 'user1';
GRANT ALL PRIVILEGES ON *.* TO 'user1'@'localhost';
```

Exit from mysql

---

Login to MySQL with user1

```
mysql -u user1 -p
```

Check if the user privileges are working.
Exit from mysql

---

**REVOKE**

Login to MySQL with the super user (root)

```
mysql -u root -p
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'newuser'@'localhost';
```

Exit from mysql

---

Login to MySQL with newuser
Check if the privileges are revoked.

---

Login to MySQL with the super user (root)

```
mysql -u root -p
drop user newuser@localhost;
```

---

```
REVOKE INSERT ON *.* FROM ‘user1’@'localhost';
```

Exit from mysql

---

Login to MySQL with user1
Check if the insert command is revoked for the user.

---

Date: 4-02-2022 **(Batch 1-Roll no.31 to 61)**

08-02-2022 **(Batch 2- Roll no.1 to 30)**

**To implement Functions**

Experiment No.8

**Aim:**

To implement Functions in database;

**Questions:**

Create a function to determine the salary grade of employees in employee table.

Function

```
DELIMITER $$

CREATE FUNCTION salaryGrade2(salary double) RETURNS
VARCHAR(10) DETERMINISTIC

BEGIN

DECLARE lvl varchar(10);

IF salary>=10000 THEN

SET lvl='PLATINUM';

ELSEIF salary>=5000  AND salary<10000  THEN

SET lvl='GOLD';

ELSE set lvl='SILVER';

END IF;

RETURN (lvl);

END

$$
```

Function Call

```
Select salaryGrade(salary);
```

Date: 9-02-2022 **(Batch 1-Roll no.31 to 61)**

11-02-2022 **(Batch 2- Roll no.1 to 30)**

**To implement Stored procedure**

Experiment No. 9

**Aim:**

To implement Stored procedure in database

**Questions:**

1) **Create a stored procedure to display the records of a table**
2) **Create a stored procedure which accepts an input parameter**
3) **Create a stored procedure which accepts an input parameter from the user and display the results accordingly.**

**Queries**

**1)Procedure without parameters**

```
DELIMITER $$

create procedure displayEmp()

begin

select * from emp;

end

$$
```

Procedure Call

```
call displayEmp()$$
```

**2)Procedure giving parameters (One input parameter)**

```
create procedure getEmpName(IN ename varchar(20))

begin

select * from emp1 where name=ename;

end

$$
```

Procedure Call

```
call getEmpName("Reshma")$$
```

**3)Procedure giving input parameter and getting output parameter**

```
create procedure getEmpCount(IN salary float, OUT empCount
int)

 begin

select count(*) into empCount from emp1 where sal>=salary;

end $$
```

## Procedure Call

```
call getEmpCount(25000,@empCount)$$

select @empCount$$
```

Date: 9-02-2022 **(Batch 1-Roll no.31 to 61)**

11-02-2022 **(Batch 2- Roll no.1 to 30)**

To implement triggers

Experiment No. 10

**Aim:**

To implement triggers in database.

Questions

1) Create a trigger which gets activated before data is inserted into the table
2) Create a trigger which gets activated before data in the table is updated.

Queries

1)

```
CREATE TABLE newAccount(acct_num INT, amount DECIMAL(10,2));
-----------------------------------------------------------
delimiter $$ ;

CREATE TRIGGER ins_sum_acc BEFORE INSERT ON newAccount FOR
EACH ROW SET @sum = @sum + NEW.amount;

SET @sum = 0;

INSERT INTO newAccount VALUES(1311,60000);
-------------------------------------------------------------
SELECT @sum AS 'Total amount inserted';

SELECT @sum$$
-------------------------------------------------------------
CREATE TRIGGER ins_transaction BEFORE INSERT ON newAccount FOR
EACH ROW PRECEDES ins_sum_acc

SET @deposits = @deposits + IF(NEW.amount>0,NEW.amount,0),
@withdrawals = @withdrawals + IF(NEW.amount<0,-NEW.amount,0);
$$
-------------------------------------------------------------S
ET @deposits=0;$$

SET @withdrawals=0;$$
```

```
---------------------------------------------------------------

SELECT @deposits;$$

SELECT @withdrawals;$$
---------------------------------------------------------------
```

2)

```
CREATE TABLE employees(id INT AUTO_INCREMENT PRIMARY KEY,
employeeNumber INT NOT NULL, lastname VARCHAR(50) NOT NULL);


CREATE TABLE employees_audit ( id INT AUTO_INCREMENT PRIMARY
KEY, employeeNumber INT NOT NULL, lastname VARCHAR(50) NOT
NULL, changedat DATETIME DEFAULT NULL, action VARCHAR(50)
DEFAULT NULL );

DELIMITER $$

CREATE TRIGGER before_employee_update BEFORE UPDATE ON
employees FOR EACH ROW BEGIN INSERT INTO employees_audit SET
action = 'update', employeeNumber = OLD.employeeNumber,
lastname = OLD.lastname, changedat = NOW(); END$$ DELIMITER ;
---------------------------------------------------------------
```

Date: 14-02-2022 **(Batch 1-Roll no.31 to 61)**

15-02-2022 **(Batch 2- Roll no.1 to 30)**

To implement cursors

Experiment No. 11

**Aim:**

To implement cursors in database.

Questions

1) Create a cursor associated with a select statement in a table and fetch each record using loop.

Queries

```
DELIMITER $$

CREATE PROCEDURE build_email_list (INOUT email_list
varchar(4000))

BEGIN

DECLARE v_finished INTEGER DEFAULT 0;

DECLARE v_email varchar(100) DEFAULT "";

-- declare cursor for employee email

DECLARE email_cursor CURSOR FOR

SELECT email FROM employees;

-- declare NOT FOUND handler

DECLARE CONTINUE HANDLER

FOR NOT FOUND SET v_finished = 1;

OPEN email_cursor;

get_email: LOOP

FETCH email_cursor INTO v_email;

IF v_finished = 1 THEN

LEAVE get_email;

END IF;

-- build email list

SET email_list = CONCAT(v_email,";",email_list);
```

```
END LOOP get_email;

CLOSE email_cursor;

END$$

DELIMITER ;

-------------------------------------------------------------

SET @email_list = "";

CALL build_email_list(@email_list);

SELECT @email_list
```

Date: 14-02-2022 **(Batch 1-Roll no.31 to 61)**

15-02-2022 **(Batch 2- Roll no.1 to 30)**

To implement various control structures using PL/SQL

Experiment No. 12

**Aim:**

To implement various control structures like IF-THEN, IF-THEN-ELSE, IF THEN ELSE IF, CASE, WHILE using PL/SQL

**Questions**

1) Create a stored procedure with IF control structure to find the largest of two numbers
2) Create a stored procedure with IF control structure to find the largest of three numbers
3) Create a stored function using CASE
4) Create a stored procedure using WHILE

**Queries**

```
1) DELIMITER $$
CREATE PROCEDURE findLargest(IN NUM1 INT, IN NUM2 INT,
OUT LARGEST INT)
BEGIN
IF NUM1 > NUM2 THEN
SET LARGEST = NUM1;
ELSE
SET LARGEST = NUM2;
END IF;
END $$
--------------------------------------------------------
CALL findLargest(10,12,@LARGEST);$$

SELECT @LARGEST;

--------------------------------------------------------
```

```
2) DELIMITER $$
```

```
CREATE PROCEDURE findLargestOfThree(IN NUM1 INT, IN NUM2
INT, IN NUM3 INT ,OUT LARGEST INT)
BEGIN
IF NUM1 > NUM2 AND NUM1 > NUM3 THEN
SET LARGEST = NUM1;
ELSE IF NUM2 > NUM1 AND NUM2 > NUM3 THEN
SET LARGEST = NUM2;
ELSE
SET LARGEST = NUM3;
END IF;
END IF;
END $$
```

---

```
CALL findLargestOfThree(10,12,13,@LARGEST);$$

SELECT @LARGEST;
```

---

3)

```
DELIMITER $$
CREATE FUNCTION findGrade2(rno int)RETURNS varchar(20)
DETERMINISTIC
BEGIN
DECLARE grade varchar(20);
DECLARE m decimal(10,2);
SELECT marks INTO m FROM student WHERE rollno=rno;
CASE
WHEN m>90 THEN SET grade=”O”;
WHEN m>80 THEN SET grade=”A”;
WHEN m>70 THEN SET grade=”B”;
WHEN m>60 THEN SET grade=”C”;
ELSE SET grade=”F”;
END CASE;
RETURN (grade);
END $$;
```

---

```
select findGrade1(124);
```

-------------------------------------------------------

4) DELIMITER $$

```
CREATE PROCEDURE REVERSE(IN NUM INT, OUT REV INT)

BEGIN

DECLARE R INT;

SET R = 0;

WHILE NUM>0 DO

SET R=(R*10)+(NUM%10);
SET NUM=NUM/10;
END WHILE;
SET REV=R;
END $$
---------------------------------------------------
CALL REVERSE1(234234234,@REV);$$
SELECT @REV;
```

---

## Packages - Theory

A package is a schema object that groups logically related PL/SQL types, variables, constants, subprograms, cursors, and exceptions. A package is compiled and stored in the database, where many applications can share its contents.

Date: 18-02-2022 **(Batch 1-Roll no.31 to 61)**

22-02-2022 **(Batch 2- Roll no.1 to 30)**

To implement packages

Experiment No. 13

**Aim:**

To implement packages in SQL

**Questions**

1) **Create a package which consists of a procedure to display the bonus of employees.**

**Queries**

```
create table employees(empid number, hire_date
date,bonus decimal);

insert into employees
values(100,'10-Jan-2010',8000);

insert into employees
values(100,'10-Feb-2010',9000);

insert into employees
values(100,'10-Mar-2010',7000);

select * from employees;
-----------------------------------------------
CREATE PACKAGE emp_bonus_resh1 AS

  PROCEDURE calc_bonus (date_hired
employees.hire_date%TYPE, );

END emp_bonus_resh1;

/
-----------------------------------------------
```

```
CREATE OR REPLACE PACKAGE BODY emp_bonus_resh1
AS
  PROCEDURE calc_bonus
    (date_hired employees.hire_date%TYPE) IS
    bonus_resh number;
  BEGIN
  select bonus into bonus_resh from employees
where hire_date=date_hired;
    DBMS_OUTPUT.PUT_LINE
      ('Employees hired on ' || date_hired || '
get bonus'|| bonus_resh);
  END;
END emp_bonus_resh1;
/
----------------------------------------------
EXEC emp_bonus_resh1.calc_bonus('10-Mar-2010');
EXEC emp_bonus_resh1.calc_bonus('10-Jan-2010');
EXEC emp_bonus_resh1.calc_bonus('10-Feb-2010');
```

Date: 18-02-2022 **(Batch 1-Roll no.31 to 61)**

22-02-2022 **(Batch 2- Roll no.1 to 30)**

To create PL/SQL blocks for Exception Handling

Experiment No. 14

**Aim:**

To create PL/SQL blocks for Exception Handling

**Questions**

1) **Create a procedure and illustrate divide by zero exception handling in PL/SQL**

**Queries**

```
CREATE PROCEDURE print_reciprocal (n NUMBER)
AUTHID DEFINER IS

BEGIN

  BEGIN

    DBMS_OUTPUT.PUT_LINE(1/n);

  EXCEPTION

    WHEN ZERO_DIVIDE THEN

      DBMS_OUTPUT.PUT_LINE('Error in inner
block:');

      DBMS_OUTPUT.PUT_LINE(1/n || ' is
undefined.');

  END;

EXCEPTION
```

```
  WHEN ZERO_DIVIDE THEN  -- handles exception
raised in exception handler
    DBMS_OUTPUT.PUT('Error in outer block: ');
    DBMS_OUTPUT.PUT_LINE('1/0 is undefined.');
END;
/
 --------------------------------------------
BEGIN
  print_reciprocal(0);
END;
/
```

Date: 25-02-2022 **(Batch 1-Roll no.31 to 61)**

26-02-2022 **(Batch 2- Roll no.1 to 30)**

To import and export data to and from a database.

Experiment No. 15

**Aim:**

To import data to a database table from a txt or csv file and export from a database to a txt or csv file

**Questions**

1) **Import data to employee table from a csv file and a text file**
2) **Export data from employee table to a csv file and a text file**

**Queries**

```
    1)  select * from emp into outfile "path\
        reshOutFile1.csv" FIELDS TERMINATED BY
        ',';
  ----------------------------------------------
--------------------------------------
  select * from emp into outfile "path\
reshOutFile2.txt" FIELDS TERMINATED BY ',';
  ----------------------------------------------
--------------------------------------
1)  LOAD DATA INFILE 'path\reshOutFile1.csv'
  INTO TABLE emp1 FIELDS TERMINATED BY ',';
```

---

```
  LOAD DATA INFILE 'path\reshOutFile2.txt' INTO
  TABLE emp FIELDS TERMINATED BY ',';
```

---

Date: 25-02-2022 **(Batch 1-Roll no.31 to 61)**

26-02-2022 **(Batch 2- Roll no.1 to 30)**

To design a database schema for an application with ER diagram from a problem description and familiarization of a database User Interface (UI).

Experiment No. 16

**Aim:**

- To design a database schema for an application with ER diagram from a problem description.
- Familiarization of MySQL Workbench (UI)
    - o Creation, configuration and deletion of databases,
    - o Creation of database schema
    - o Export ER diagram from a database schema
    - o Bulk import and export to and from the database respectively

**Questions**

1) To design a database schema for the application (University Management System) with ER diagram.
2) MySQL Workbench operations

(Left hand side)

1) ER Diagram
2) Write the menu options for the  (No need to draw results just write steps)
    a. creation of database
    b. creation of tables
    c. Export ER diagram
    d. Import and Export

**Result**

Designed a database schema for University Management System with ER diagram and all the operations are performed in MySQL Workbench.

## Inference

Designed database schema for University Management System and familiarized with MySQL Workbench UI

Date: 28-02-2022 **(Batch 1-Roll no.31 to 61)**

28-02-2022 **(Batch 2- Roll no.1 to 30)**

To familiarize with NoSQL databases and Create, Read, Update and Delete (CRUD) operations.

Experiment No. 17

**Aim:**

To familiarize with MongoDB (NoSQL database) and its CRUD operations.

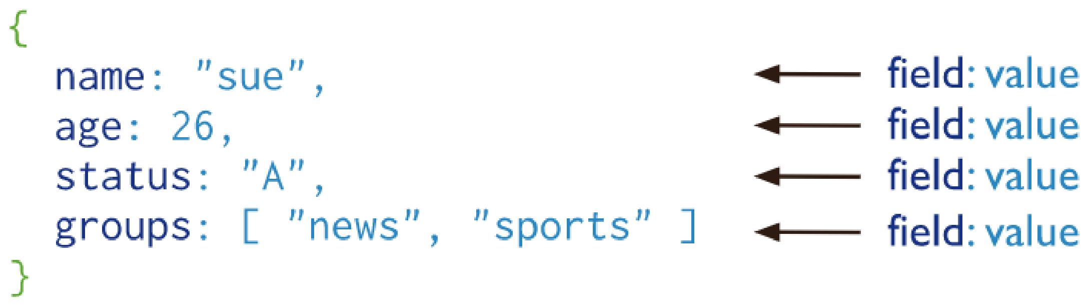
**Theory:**

**Introduction to MongoDB (NoSQL database)**

MongoDB, one of the most popular NoSQL database management system, is an open-source document-oriented database. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information. MongoDB is a document database used to build highly available and scalable internet applications. With its flexible schema approach, it's popular with development teams using agile methodologies.

**Document Database**

Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. MongoDB stores documents in collections. Collections are analogous to tables in relational databases. A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.



```
{
   name: "sue",
   age: 26,
   status: "A",
   groups: [ "news", "sports" ]
}
```

**MongoDB CRUD Operations**

CRUD operations (*create*, *read*, *update*, and *delete)* documents.

**Create Operations**

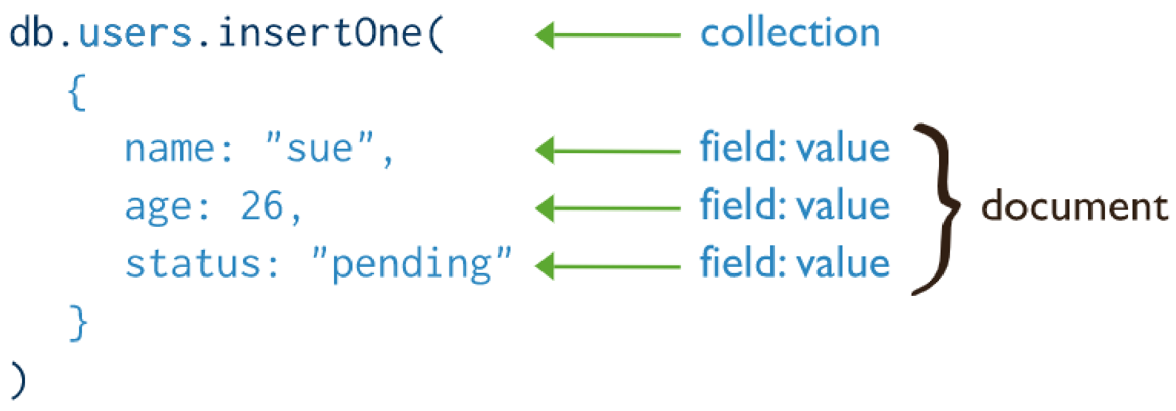
Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

- db.collection.insertOne()
- db.collection.insertMany()

  In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

Left hand side



```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

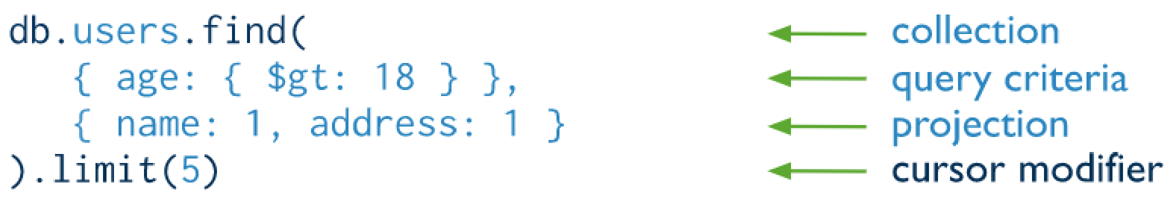
**Read Operations**

Read operations retrieve documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

- db.collection.find()

We can specify query filters or criteria that identify the documents to return.

Left hand side



```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

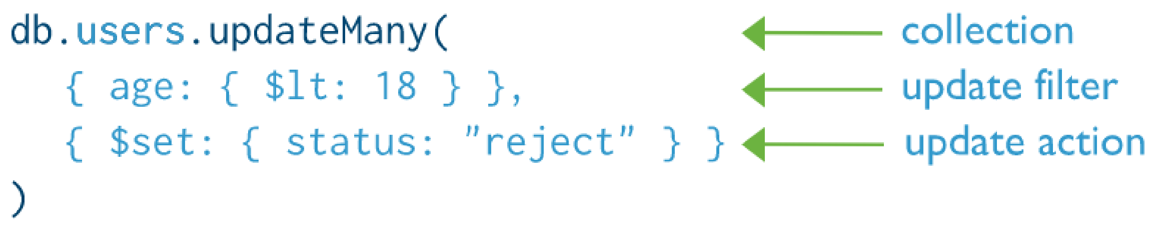
## Update Operations

Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

    In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document. We can specify criteria, or filters, that identify the documents to update. These filters use the same syntax as read operations.

Left hand side



```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

## Delete Operations

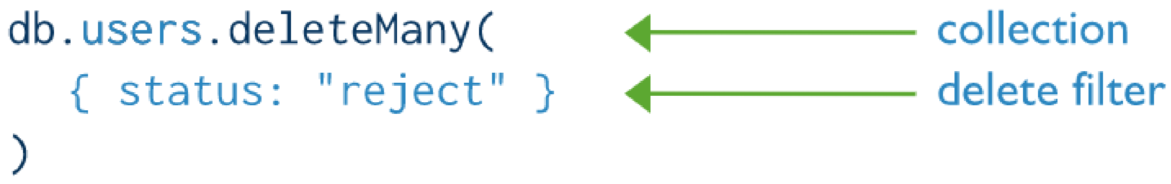
Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

- db.collection.deleteOne()
- db.collection.deleteMany()

In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

We can specify criteria, or filters, that identify the documents to remove. These filters use the same syntax as read operations.

Left hand side



```
db.users.deleteMany(
   { status: "reject" }
)
```

**Inference**

Familiarized with MongoDB and its CRUD operations

Date: -03-2022 **(Batch 1-Roll no.31 to 61)**

-03-2022 **(Batch 2- Roll no.1 to 30)**

To design a database application using any front-end tool for any problem selected

Experiment No. 18

**Aim:**

In case of any error,

```
        show variables like 'secure%';
```

```
mysql> show variables like 'secure%';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| secure_file_priv |       |
+------------------+-------+
```

Here is what worked for me in Windows 7 to disable secure-file-priv

1. Stop the MySQL server service by going into services.msc.
2. Go to C:\ProgramData\MySQL\MySQL Server 5.6
3. Open the my.ini file in Notepad.
4. Search for 'secure-file-priv'.
5. Comment the line out by adding '#' at the start of the line. For MySQL Server 5.7.16 and above, commenting won't work. You have to set it to an empty string like this one - secure-file-priv=""
6. Save the file.
7. Start the MySQL server service by going into services.msc.