SQL Queries: Basic to Advanced Level

1) Create a Database 'Classroom'.

Ans:

CREATE DATABASE Classroom;

2) Create a table named 'Science_class' with the following properties 3 Cloumn (Enrollment_no INT, Name VARCHAR, Science_Marks INT)

Ans:

USE Classroom;

CREATE TABLE Science_class (Enrollment_no INT, Name VARCHAR(50), Science_Marks INT);

3) Insert the following data into Science_class using insert into command

1 Popeye 33

2 Olive 54

3 Brutus 98

ans:

USE Classroom;

INSERT INTO Science_class (Enrollment_no, Name, Science_Marks) VALUES (1, 'Popeye', 33);

INSERT INTO Science_class (Enrollment_no, Name, Science_Marks) VALUES (2, 'Olive', 54);

INSERT INTO Science_class (Enrollment_no, Name, Science_Marks) VALUES (3, 'Brutus', 98);

4) Retrieve all data from the table 'Science_Class'

ans :

SELECT * FROM Science_class;

5) Retrieve the name of students who have scored more than 60 marks

ans :

SELECT Name FROM Science_class WHERE Science_Marks > 60;

6) Retrieve all data of students who have scored more than 35 but less than 60 marks

ans :

SELECT * FROM Science_class WHERE Science_Marks > 35 AND Science_Marks < 60;

7) Retrieve all other students i.e. who have scored less than or equal to 35 or more than or equal to 60.

ans :

SELECT * FROM Science_class WHERE Science_Marks <= 35 OR Science_Marks >= 60;

8) Update the marks of Popeye to 45

ans :

UPDATE Science_class SET Science_Marks = 45 WHERE Name = 'Popeye';

9) Delete the row containing details of student named 'Robb'

ans :

DELETE FROM Science_class WHERE Name = 'Robb';

10) Rename column 'Name' to 'student_name'.

ans :

ALTER TABLE Science_class RENAME COLUMN Name TO student_name;

11) Backup this database into a TAR file

ans :

mysqldump -u (username) -p Classroom > Classroom_backup.sql

tar -cvzf Classroom_backup.tar.gz Classroom_backup.sql

The first command creates a SQL dump file of the 'Classroom' database named 'Classroom_backup.sql', while the second command creates a TAR file named 'Classroom_backup.tar.gz' that contains the SQL dump file.

12) Drop the 'science_class' table

ans :

DROP TABLE Science_class;

13) Restore from the backup file to get back the deleted table

ans :

tar -xzvf Classroom_backup.tar.gz

mysql -u (username) -p Classroom < Classroom_backup.sql

The first command extracts the SQL dump file from the TAR file, while the second command restores the 'Classroom' database from the SQL dump file. After restoring, the 'Science_class' table will be back in the 'Classroom' database with all its previous data.

I am using another database that contains tables for Customers, Products, and Sales to learn advanced SQL queries

14) Get the list of all cities where the region is north or east without any duplicates using IN statement

ans :

```
SELECT DISTINCT City FROM customer_details WHERE Region IN ('north', 'east');
```

This query will retrieve all the distinct cities from the 'customer_details' table where the region is either 'north' or 'east'.

15)Get the list of all orders where the 'sales' value is between 100 and 500 using the BETWEEN operator

ans :

```
SELECT * FROM order_details WHERE sales BETWEEN 100 AND 500;
```

This query will retrieve all the orders from the 'order_details' table where the 'sales' value is between 100 and 500 (inclusive)

16)Retrieve all orders where 'discount' value is greater than zero ordered in descending order basis 'discount' value

ans :

```
SELECT * FROM order_details WHERE discount > 0 ORDER BY discount DESC;
```

This query will retrieve all the orders from the 'order_details' table where the 'discount' value is greater than zero, and order the results in descending order based on the 'discount' value.

17)Limit the number of results in above query to top 10

ans :

```
SELECT * FROM order_details WHERE discount > 0 ORDER BY discount DESC LIMIT 10;
```

18)Find the sum of all 'sales' values.

ans :

SELECT SUM(sales) AS total_sales FROM order_details;

19) Find count of the number of customers in north region with age between 20 and 30

ans :

SELECT COUNT(*) AS customer_count FROM customer_details WHERE Region = 'north' AND age BETWEEN 20 AND 30;

This query will retrieve the count of customers from the 'customer_details' table who belong to the north region and whose age is between 20 and 30, and alias the result as 'customer_count'.

20)Find the average age of East region customers

ans :

SELECT AVG(age) AS avg_age FROM customer_details WHERE Region = 'east';

21)Find the Minimum and Maximum aged customer from Philadelphia

ans :

SELECT MIN(age) AS min_age, MAX(age) AS max_age FROM customer_details WHERE City = 'Philadelphia';

22)Get the list of product ID's where the quantity of product sold is greater than 10

ans :

SELECT product_id FROM orders WHERE quantity > 10 GROUP BY product_id;

23) Get data containing Product_id, product name, category, total sales value of that product and total quantity sold. (Use sales and product table)

ans :

SELECT p.product_id, p.product_name, p.category, SUM(s.sales) AS total_sales, SUM(s.quantity) AS total_quantity_sold FROM product p JOIN sales s ON p.product_id = s.product_id GROUP BY p.product_id, p.product_name, p.category;

This query will retrieve the product ID, product name, category, total sales value and total quantity sold for each product from the 'product' table and 'sales' table. The results will be grouped by product ID, product name and category.

24) Create a View which contains order_line, Product_id, sales and discount value of the first order date in the sales table and name it as "Daily_Billing"

ans :

CREATE VIEW Daily_Billing AS SELECT order_line, product_id, sales, discount FROM sales WHERE order_date = (SELECT MIN(order_date) FROM sales);

This query creates a view that selects the order_line, product_id, sales, and discount columns from the 'sales' table where the order_date matches the minimum order_date in the 'sales' table. The view will contain the data from the first order date.

25)To delete the "Daily_Billing" view, use the following SQL query

ans :

DROP VIEW IF EXISTS Daily_Billing;

This query will delete the "Daily_Billing" view if it exists.

26)Find Maximum length of characters in the Product name string from Product table

ans :

SELECT MAX(LENGTH(product_name)) AS max_length FROM Product;

27)Retrieve product name, sub-category and category from Product table and an additional column named "product_details" which contains a concatenated string of product name, sub-category and category

ans :

SELECT product_name, sub_category, category, CONCAT(product_name, ' - ', sub_category, ' - ', category) AS product_details FROM Product;

This query uses the CONCAT function to concatenate the 'product_name', 'sub_category', and 'category' columns into a new column named "product_details".

28)List down comma separated product name where sub-category is either Chairs or Tables

ans :

SELECT GROUP_CONCAT(product_name SEPARATOR ', ') AS product_names FROM Product WHERE sub_category IN ('Chairs', 'Tables');

This query uses the GROUP_CONCAT function to concatenate the 'product_name' column into a comma-separated list and displays it as 'product_names' where the 'sub_category' column matches either 'Chairs' or 'Tables'.

29)You are running a lottery for your customers. So, pick a list of 5 Lucky customers from customer table using random function

ans :

SELECT * FROM customer ORDER BY RAND() LIMIT 5;

This will select all columns from the customer table and order the result set randomly using the RAND() function. The LIMIT clause limits the result set to 5 rows.

30) Find out all customers who have first name and last name of 5 characters each and last name starts with "a/b/c/d"

ans :

```
SELECT * FROM customers WHERE LENGTH(first_name) = 5 AND LENGTH(last_name) = 5 AND
last_name REGEXP '^[abcd]'
```

This query will retrieve all customers from the "customers" table whose first name and last name have 5 characters each and last name starts with either 'a', 'b', 'c', or 'd'.

In [ ]: