

**blinkit**

## Introduction:-

**Blinkit** (formerly known as **Grofers**) is an Indian online grocery delivery service that allows customers to purchase groceries, daily essentials, snacks, and other products online. The platform was founded in 2013 by **Albinder Dhindsa** and **Saurabh Kumar** with the aim of simplifying and streamlining the process of grocery shopping. Blinkit offers a wide range of products, including fresh fruits and vegetables, dairy products, packaged foods, and personal care items.

### **Key Features of Blinkit:**

1. **Fast Delivery:** Blinkit is known for its quick delivery service, often promising deliveries in as little as 10-30 minutes depending on the location. This speed sets it apart from traditional e-commerce platforms that generally have longer delivery timelines.
2. **Variety of Products:** Blinkit offers a vast range of products across multiple categories such as groceries, household products, personal care, baby care, and more. It also provides an assortment of health and wellness products.
3. **User-Friendly App:** Blinkit operates through a mobile application and a website, making it easy for customers to browse through available products, place orders, and track deliveries. The platform is designed to provide a seamless user experience.
4. **Fresh Produce:** One of the standout features of Blinkit is its emphasis on delivering fresh fruits, vegetables, and other perishable goods. This has become a key differentiator from other online grocery platforms.
5. **Subscription and Offers:** Blinkit often provides subscription-based models for regular purchases and offers various discounts, cashback, and deals to attract customers. The platform also runs promotional campaigns to enhance customer engagement.
6. **Wide Network of Warehouses:** Blinkit operates through a network of warehouses strategically placed across different regions. These warehouses help facilitate quick delivery and allow for efficient inventory management.

## **Applications of the project:-**

### **1) Customer Segmentation Analysis:**

- Identify **top customer segments** based on total orders and average order value.
- Find **high-value customers** who place frequent or large orders.

### **2) Product Insights:-**

- Identify products with the **highest margin percentage** (profitability).
- Compare **MRP vs. selling price** to analyze discounts and pricing strategy.

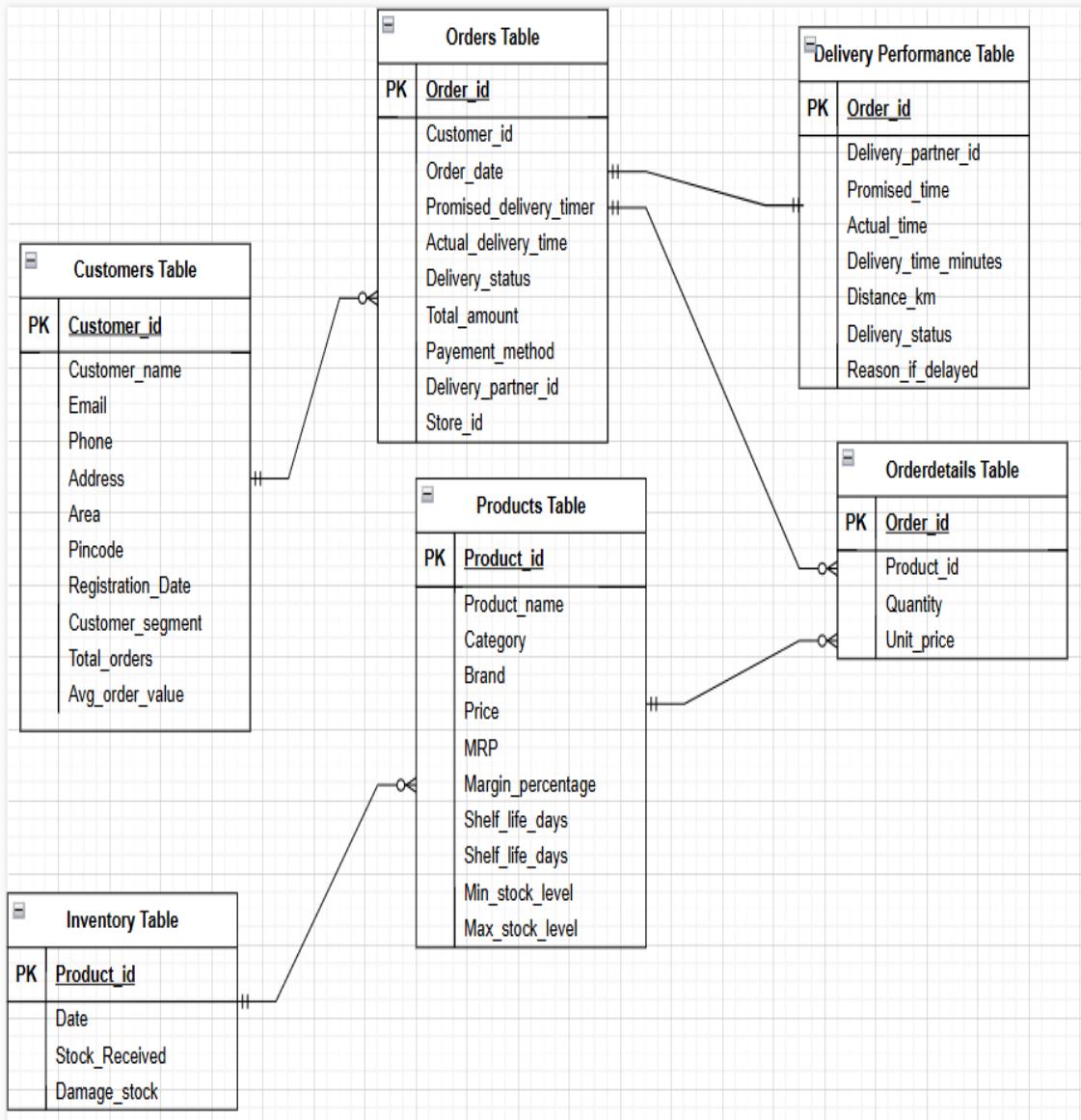
### **3) Inventory analysis:-**

- Monitor stock trends over **months** to identify seasonal patterns or operational inefficiencies.

### **4) Revenue and Sales Reports:-**

- Track **sales trends over time** to find peak days.

# ER Diagram:-



## Blinkit- Quick-Commerce Platform Database Design

Database- Blinkit

Tables-

- Blinkit Customer Table
- Blinkit Products Table
- Blinkit Orders Detail Table

- Blinkit Order Table
- Blinkit Inventory Table
- Blinkit Delivery Performance Table

### **Key Relationships:**

1. **Customers → Orders (1:M)**
2. **Orders → Order Details (1:M)**
3. **Products → Order Details (1:M)**
4. **Products → Inventory (1:1 or 1:M)**
5. **Orders → Delivery Performance (1:1)**

The screenshot shows the SSMS interface with the following details:

- Navigator:** Shows the "MANAGEMENT" and "INSTANCE" sections.
- Query Editor:** Titled "Query 1" and "SQL File 3\*". It contains the following SQL command:

```
CREATE DATABASE Blinkit;
```
- Output Window:** Titled "Action Output". It displays the execution log:

#	Time	Action
1	01:42:17	CREATE DATABASE Blinkit

## DATA DEFINITION LANGUAGE (DDL):

### 1){Create}-

- o Creating Database and Tables

#### A) Create Database Blinkit:-

#### B) Using the Database:-

The screenshot shows the SSMS interface with the following details:

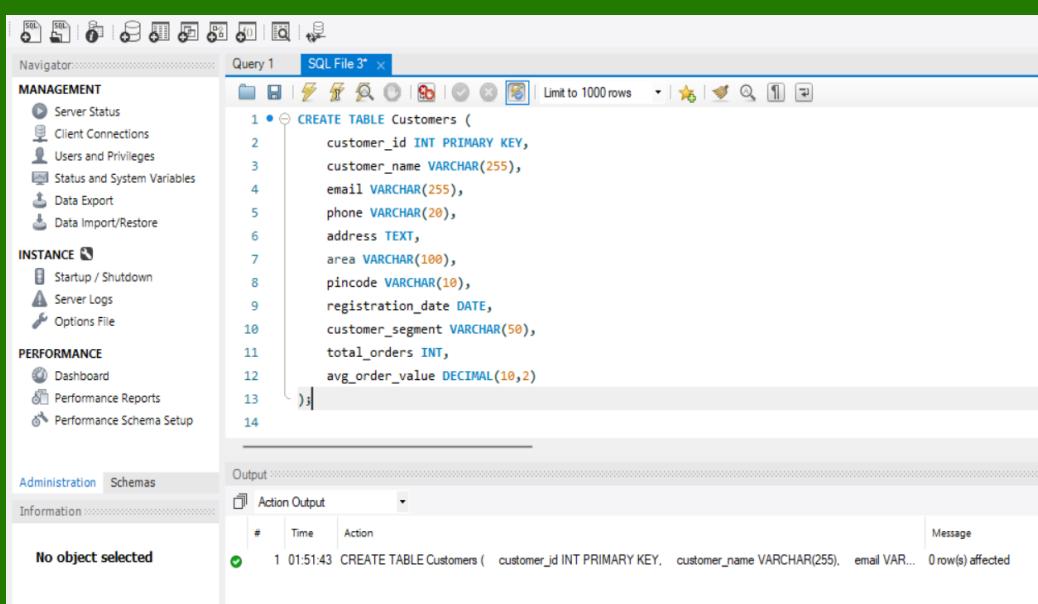
- Navigator:** Shows the "MANAGEMENT" and "INSTANCE" sections.
- Query Editor:** Titled "Query 1" and "SQL File 3\*". It contains the following SQL command:

```
USE Blinkit;
```
- Output Window:** Titled "Action Output". It displays the execution log:

#	Time	Action
1	01:47:46	USE Blinkit

#### C) Creating Table Customer:-

#### D) Creating Table Products:-



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** On the left, under the "MANAGEMENT" section, "Server Status", "Client Connections", "Users and Privileges", "Status and System Variables", "Data Export", and "Data Import/Restore" are listed.
- Query Editor:** The main window displays a SQL query for creating a "Customers" table:

```
1 • ⚡ CREATE TABLE Customers (
2     customer_id INT PRIMARY KEY,
3     customer_name VARCHAR(255),
4     email VARCHAR(255),
5     phone VARCHAR(20),
6     address TEXT,
7     area VARCHAR(100),
8     pincode VARCHAR(10),
9     registration_date DATE,
10    customer_segment VARCHAR(50),
11    total_orders INT,
12    avg_order_value DECIMAL(10,2)
13 )|
```
- Output:** Below the query editor, the "Action Output" tab shows the execution log:

#	Time	Action	Message
1	01:51:43	CREATE TABLE Customers ( customer_id INT PRIMARY KEY, customer_name VARCHAR(255), email VAR... )	0 row(s) affected
- Status Bar:** At the bottom, it says "No object selected".

The screenshot shows the MySQL Workbench interface with the 'Query 1' tab active. The code pane contains the following SQL query:

```

1 • ⚡ CREATE TABLE Products (
2     product_id INT PRIMARY KEY,
3     product_name VARCHAR(255),
4     category VARCHAR(100),
5     brand VARCHAR(100),
6     price DECIMAL(10,2),
7     mrp DECIMAL(10,2),
8     margin_percentage DECIMAL(5,2),
9     shelf_life_days INT,
10    min_stock_level INT,
11    max_stock_level INT
12 );
13
14

```

The output pane shows the execution log:

#	Time	Action	Message
1	01:55:06	CREATE TABLE Products ( product_id INT PRIMARY KEY, product_name VARCHAR(255), category VARC... )	0 row(s) affected

## E) Creating Table Orders:-

The screenshot shows the MySQL Workbench interface with the 'Query 1' tab active. The code pane contains the following SQL query:

```

1 • ⚡ CREATE TABLE Orders (
2     order_id INT PRIMARY KEY,
3     customer_id INT,
4     order_date DATETIME,
5     promised_delivery_time DATETIME,
6     actual_delivery_time DATETIME,
7     delivery_status VARCHAR(50),
8     order_total DECIMAL(10,2),
9     payment_method VARCHAR(50),
10    delivery_partner_id INT,
11    store_id INT,
12    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
13 );
14

```

The output pane shows the execution log:

#	Time	Action	Message
1	02:20:38	CREATE TABLE Orders ( order_id INT PRIMARY KEY, customer_id INT, order_date DATETIME, promise... )	0 row(s) affected

## F) Creating Table Delivery Performance:-

The screenshot shows the MySQL Workbench interface with the 'Query 1' tab active. The code pane contains the following SQL query:

```

1 • ⚡ CREATE TABLE DeliveryPerformance (
2     order_id INT PRIMARY KEY,
3     delivery_partner_id INT,
4     promised_time DATETIME,
5     actual_time DATETIME,
6     delivery_time_minutes INT,
7     distance_km DECIMAL(5,2),
8     delivery_status VARCHAR(50),
9     reasons_if_delayed TEXT,
10    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
11 );
12

```

The output pane shows the execution log:

#	Time	Action	Message
1	02:24:49	CREATE TABLE DeliveryPerformance ( order_id INT PRIMARY KEY, delivery_partner_id INT, promised_time... )	0 row(s) affected

## G) Creating Table Order Details:-

The screenshot shows the MySQL Workbench interface with the 'Query 1' tab selected. The code pane contains the SQL command to create the 'OrderDetails' table:

```
CREATE TABLE OrderDetails (
    order_id INT,
    product_id INT,
    quantity INT,
    unit_price DECIMAL(10,2),
    PRIMARY KEY (order_id, product_id),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

The output pane shows the execution log:

#	Time	Action	Message
1	02:29:00	CREATE TABLE OrderDetails ( order_id INT, product_id INT, quantity INT, unit_price DECIMAL(10,2), ... )	0 row(s) affected

## H) Creating Table Inventory:-

The screenshot shows the MySQL Workbench interface with the 'Query 1' tab selected. The code pane contains the SQL command to create the 'Inventory' table:

```
CREATE TABLE Inventory (
    product_id INT,
    date DATE,
    stock_received INT,
    damaged_stock INT,
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

The output pane shows the execution log:

#	Time	Action	Message
1	02:30:45	CREATE TABLE Inventory ( product_id INT, date DATE, stock_received INT, damaged_stock INT, FO... )	0 row(s) affected

## 2) CHECK THE STRUCTURE OF TABLE

### A) Customers Table:-

The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected. A query window titled 'Query 1' displays the command 'desc Customers;'. The results grid shows the following table structure:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
customer_name	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
phone	varchar(20)	YES		NULL	
address	text	YES		NULL	
area	varchar(100)	YES		NULL	
pincode	varchar(10)	YES		NULL	
registration_date	date	YES		NULL	
customer_segment	varchar(50)	YES		NULL	

The message area at the bottom right indicates '11 row(s) returned'.

### B) Product Table:-

The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected. A query window titled 'Query 1' displays the command 'desc Products;'. The results grid shows the following table structure:

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	
product_name	varchar(255)	YES		NULL	
category	varchar(100)	YES		NULL	
brand	varchar(100)	YES		NULL	
price	decimal(10,2)	YES		NULL	
mrp	decimal(10,2)	YES		NULL	
margin_percentage	decimal(5,2)	YES		NULL	
shelf_life_days	int	YES		NULL	
min_stock_level	int	YES		NULL	

The message area at the bottom right indicates '10 row(s) returned'.

### C) Order Table:-

The screenshot shows the MySQL Workbench interface with the 'Schemas' tab selected. A query window titled 'Query 1' displays the command 'desc Orders;'. The results grid shows the following table structure:

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
order_date	datetime	YES		NULL	
promised_delivery_time	datetime	YES		NULL	
actual_delivery_time	datetime	YES		NULL	
delivery_status	varchar(50)	YES		NULL	
order_total	decimal(10,2)	YES		NULL	
payment_method	varchar(50)	YES		NULL	
delivery_partner_id	int	YES		NULL	

The message area at the bottom right indicates '10 row(s) returned'.

#### D) Delivery Performance Table:-

The screenshot shows the MySQL Workbench interface with the 'Query 1' tab active. The query window contains the command: `desc DeliveryPerformance;`. The results grid displays the following table structure:

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
delivery_partner_id	int	YES		NULL	
promised_time	datetime	YES		NULL	
actual_time	datetime	YES		NULL	
delivery_time_minutes	int	YES		NULL	
distance_km	decimal(5,2)	YES		NULL	
delivery_status	varchar(50)	YES		NULL	
reasons_if_delayed	text	YES		NULL	

The output pane shows the message: `8 row(s) returned`.

#### E) Order Details Table:-

The screenshot shows the MySQL Workbench interface with the 'Query 1' tab active. The query window contains the command: `desc OrderDetails;`. The results grid displays the following table structure:

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
product_id	int	NO	PRI	NULL	
quantity	int	YES		NULL	
unit_price	decimal(10,2)	YES		NULL	

The output pane shows the message: `4 row(s) returned`.

#### F) Inventory Table:-

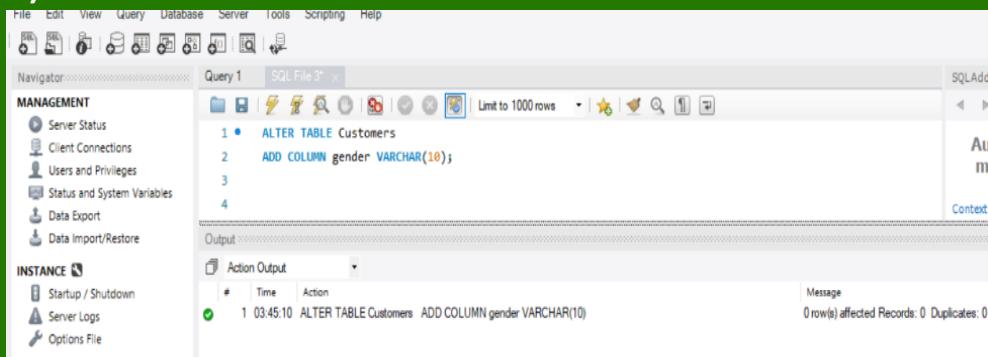
The screenshot shows the MySQL Workbench interface with the 'Query 1' tab active. The query window contains the command: `desc Inventory;`. The results grid displays the following table structure:

Field	Type	Null	Key	Default	Extra
product_id	int	YES	MUL	NULL	
date	date	YES		NULL	
stock_received	int	YES		NULL	
damaged_stock	int	YES		NULL	

The output pane shows the message: `4 row(s) returned`.

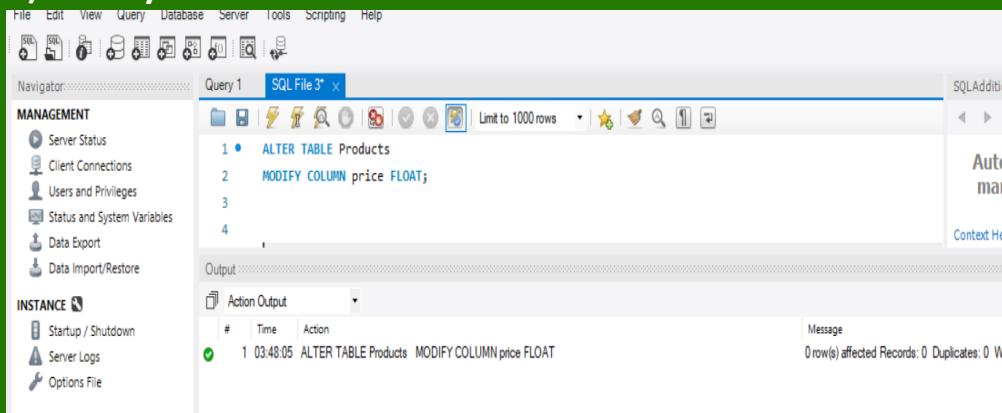
### 3) {Alter}

#### A) Add a Column:-



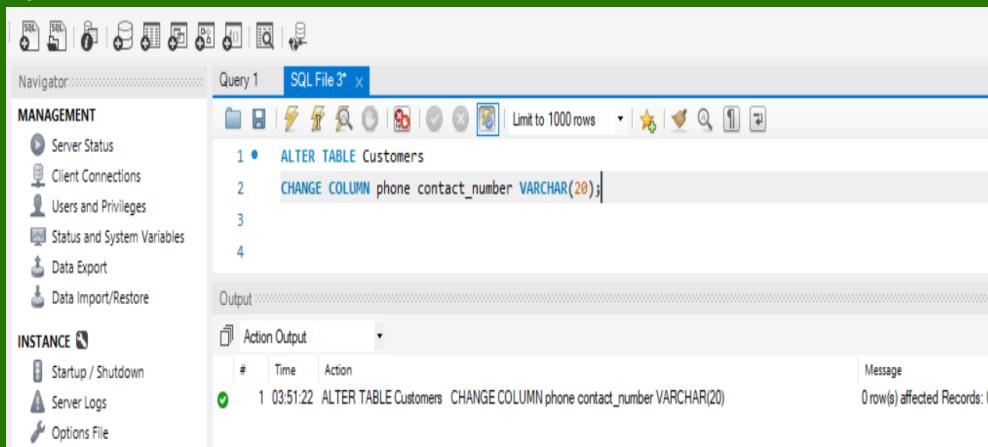
```
File Edit View Query Database Server Tools Scripting Help  
Navigator  
MANAGEMENT  
Server Status  
Client Connections  
Users and Privileges  
Status and System Variables  
Data Export  
Data Import/Restore  
INSTANCE  
Startup / Shutdown  
Server Logs  
Options File  
Query 1 SQL File 3* x  
ALTER TABLE Customers  
ADD COLUMN gender VARCHAR(10);  
Output  
Action Output # Time Action  
1 03:45:10 ALTER TABLE Customers ADD COLUMN gender VARCHAR(10)  
Message 0 row(s) affected Records: 0 Duplicates: 0
```

#### B) Modify Column:-



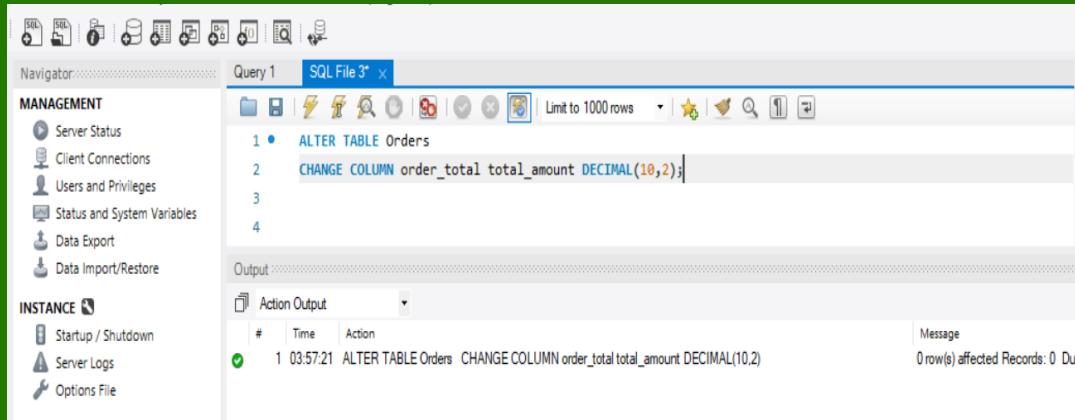
```
File Edit View Query Database Server Tools Scripting Help  
Navigator  
MANAGEMENT  
Server Status  
Client Connections  
Users and Privileges  
Status and System Variables  
Data Export  
Data Import/Restore  
INSTANCE  
Startup / Shutdown  
Server Logs  
Options File  
Query 1 SQL File 3* x  
ALTER TABLE Products  
MODIFY COLUMN price FLOAT;  
Output  
Action Output # Time Action  
1 03:48:05 ALTER TABLE Products MODIFY COLUMN price FLOAT  
Message 0 row(s) affected Records: 0 Duplicates: 0
```

#### C) Rename a Column:-



```
File Edit View Query Database Server Tools Scripting Help  
Navigator  
MANAGEMENT  
Server Status  
Client Connections  
Users and Privileges  
Status and System Variables  
Data Export  
Data Import/Restore  
INSTANCE  
Startup / Shutdown  
Server Logs  
Options File  
Query 1 SQL File 3* x  
ALTER TABLE Customers  
CHANGE COLUMN phone contact_number VARCHAR(20);  
Output  
Action Output # Time Action  
1 03:51:22 ALTER TABLE Customers CHANGE COLUMN phone contact_number VARCHAR(20)  
Message 0 row(s) affected Records: 0
```

#### D) Change a Column Name:-



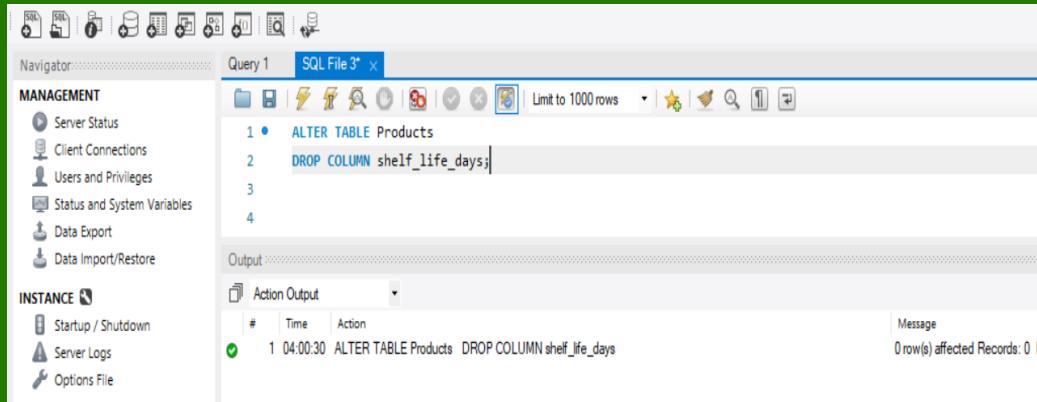
The screenshot shows the SQL Server Management Studio interface. The left pane displays the Object Explorer with nodes for MANAGEMENT and INSTANCE. The right pane contains a Query Editor window titled "Query 1" with the following SQL code:

```
1 • ALTER TABLE Orders
2     CHANGE COLUMN order_total total_amount DECIMAL(10,2);
```

The Output pane shows the execution results:

#	Time	Action	Message
1	03:57:21	ALTER TABLE Orders CHANGE COLUMN order_total total_amount DECIMAL(10,2)	0 row(s) affected Records: 0 Duration: 00:00:00.000

#### E) Drop a Column:-



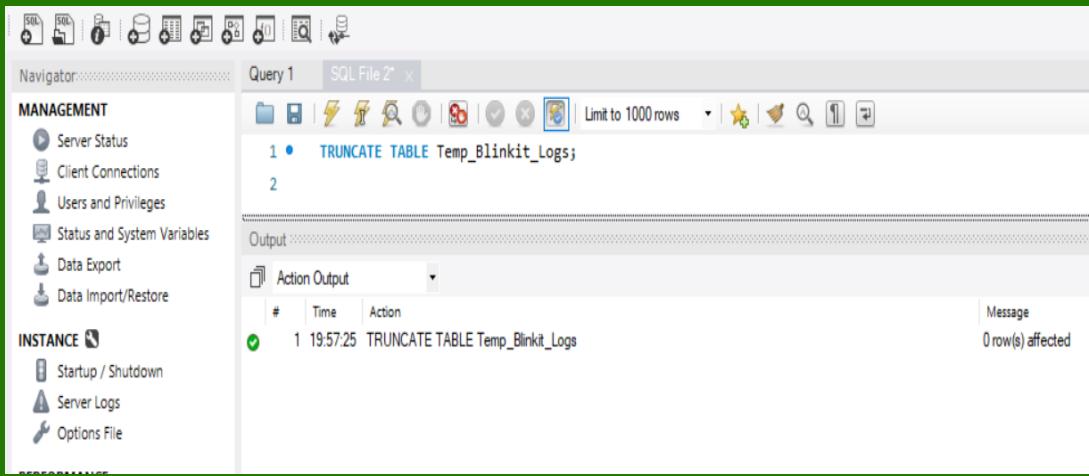
The screenshot shows the SQL Server Management Studio interface. The left pane displays the Object Explorer with nodes for MANAGEMENT and INSTANCE. The right pane contains a Query Editor window titled "Query 1" with the following SQL code:

```
1 • ALTER TABLE Products
2     DROP COLUMN shelf_life_days;
```

The Output pane shows the execution results:

#	Time	Action	Message
1	04:00:30	ALTER TABLE Products DROP COLUMN shelf_life_days	0 row(s) affected Records: 0 Duration: 00:00:00.000

#### 4) {Truncate}



The screenshot shows the SQL Server Management Studio interface. In the center, the Query Editor window titled 'Query 1' contains the following SQL command:

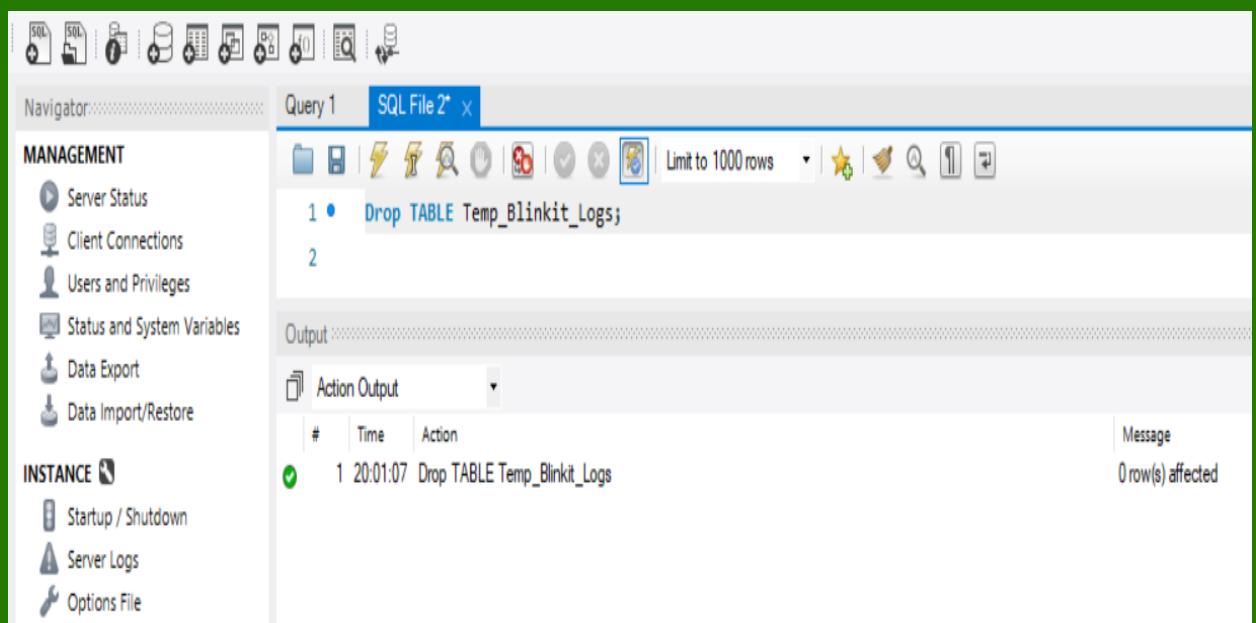
```
1 • TRUNCATE TABLE Temp_Blinkit_Logs;
```

Below the query window, the 'Output' pane displays the results of the command. It shows a table titled 'Action Output' with three columns: '#', 'Time', and 'Action'. The first row is highlighted with a green checkmark and shows the following information:

#	Time	Action
1	19:57:25	TRUNCATE TABLE Temp_Blinkit_Logs

To the right of the table, under the 'Message' column, it says '0 row(s) affected'.

#### 5) {Drop}



The screenshot shows the SQL Server Management Studio interface. In the center, the Query Editor window titled 'Query 1' contains the following SQL command:

```
1 • Drop TABLE Temp_Blinkit_Logs;
```

Below the query window, the 'Output' pane displays the results of the command. It shows a table titled 'Action Output' with three columns: '#', 'Time', and 'Action'. The first row is highlighted with a green checkmark and shows the following information:

#	Time	Action
1	20:01:07	Drop TABLE Temp_Blinkit_Logs

To the right of the table, under the 'Message' column, it says '0 row(s) affected'.

# DATA MANIPULATION LANGUAGE (DML):

## 1) {Insert Into}

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code entered is an `INSERT INTO` statement for the `Customers` table, containing 22 rows of data. The output pane shows the execution message: `1 row(s) affected Rows matched: 21 Changed: 1 Warnings: 0`.

```
1 • INSERT INTO Customers (customer_id, customer_name, email, address, area, pincode, registration_date, customer_segment, total_orders, avg_order_
2 VALUES
3 (1, 'Rahul Sharma', 'rahul.sharma@example.com', '123 MG Road, Mumbai', 'Andheri', '400053', '2023-01-15', 'Premium', 15, 650.50, 'Male'),
4 (2, 'Priya Verma', 'priya.verma@example.com', '456 Sector 21, Delhi', 'Dwarka', '110075', '2022-12-20', 'Regular', 10, 450.00, 'Female'),
5 (3, 'Amit Gupta', 'amit.gupta@example.com', '789 MG Road, Bangalore', 'Koramangala', '560034', '2023-02-10', 'Premium', 18, 720.75, 'Male'),
6 (4, 'Sneha Kapoor', 'sneha.kapoor@example.com', '159 Park Street, Kolkata', 'Salt Lake', '700091', '2023-03-05', 'Regular', 8, 320.00, 'Female'),
7 (5, 'Vikram Singh', 'vikram.singh@example.com', '357 Lajpat Nagar, Delhi', 'Lajpat Nagar', '110024', '2022-11-25', 'Premium', 22, 890.90, 'Male'),
8 (6, 'Neha Mehta', 'neha.mehta@example.com', '246 Banjara Hills, Hyderabad', 'Banjara Hills', '500034', '2023-04-12', 'Regular', 12, 480.20, 'Female'),
9 (7, 'Arjun Malhotra', 'arjun.malhotra@example.com', '951 Shivaji Park, Pune', 'Shivaji Park', '411002', '2023-05-08', 'Premium', 25, 940.00, 'Male'),
10 (8, 'Divya Nair', 'divya.nair@example.com', '753 MG Road, Chennai', 'T. Nagar', '600017', '2023-06-14', 'Regular', 9, 350.75, 'Female'),
11 (9, 'Rohit Chawla', 'rohit.chawla@example.com', '852 Sadar Bazar, Gurgaon', 'Sadar Bazar', '122001', '2023-07-10', 'Premium', 30, 1020.50, 'Male'),
12 (10, 'Anjali Joshi', 'anjali.joshi@example.com', '369 Civil Lines, Lucknow', 'Civil Lines', '226001', '2023-08-18', 'Regular', 6, 280.60, 'Female'),
13 (11, 'Sandeep Rao', 'sandeep.rao@example.com', '741 Begumpet, Hyderabad', 'Begumpet', '500015', '2023-09-05', 'Premium', 27, 950.30, 'Male'),
14 (12, 'Meera Shah', 'meera.shah@example.com', '852 Vastrapur, Ahmedabad', 'Vastrapur', '380015', '2023-10-02', 'Regular', 11, 420.80, 'Female'),
15 (13, 'Kunal Bansal', 'kunal.bansal@example.com', '951 Sector 62, Noida', 'Sector 62', '201381', '2023-11-12', 'Premium', 28, 888.00, 'Male'),
16 (14, 'Ritika Roy', 'ritika.roy@example.com', '654 Harrington, Lucknow', 'Harrington', '226003', '2023-12-15', 'Regular', 7, 310.25, 'Female'),
17 (15, 'Taru Aggarwal', 'tarun.aggarwal@example.com', '753 MG Road, Indore', 'Vijay Nagar', '452010', '2024-01-20', 'Premium', 14, 670.40, 'Male'),
18 (16, 'Pooja Iyer', 'pooja.iyer@example.com', '369 Malleshwaram, Bangalore', 'Malleshwaram', '560003', '2024-02-05', 'Regular', 5, 230.98, 'Female'),
19 (17, 'Rajesh Kumar', 'rajesh.kumar@example.com', '852 Alwarpet, Chennai', 'Alwarpet', '600018', '2024-02-20', 'Premium', 28, 970.75, 'Male'),
20 (18, 'Snehal Patil', 'snehal.patil@example.com', '654 ShivaJinagar, Pune', 'ShivaJinagar', '411005', '2024-03-10', 'Regular', 13, 530.50, 'Female'),
21 (19, 'Yashwant Deshmukh', 'yashwant.deshmukh@example.com', '159 MG Road, Bhopal', 'MP Nagar', '462011', '2024-03-25', 'Premium', 19, 788.60, 'Male'),
22 (20, 'Kavita Saxena', 'kavita.saxena@example.com', '753 Sector 10, Chandigarh', 'Sector 10', '160010', '2024-04-02', 'Regular', 4, 210.30, 'Female')
```

## 2) {Update} with SET & WHERE Clause:-

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code entered is an `UPDATE` statement for the `Customers` table, setting the `email` to a new value and specifying a condition with `WHERE customer_id = 1`. The output pane shows the execution message: `1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0`.

```
1 • UPDATE Customers
2 SET email = 'rahul.sharma_new@example.com'
3 WHERE customer_id = 1;
```

## 3) {Delete from} with Where clause:-

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code entered is a `DELETE FROM` statement for the `Customers` table, filtering by the `area` column. The output pane shows the execution message: `1 row(s) affected`, along with the deleted query text.

```
1 • DELETE FROM Customers
2 WHERE area = 'Andheri';
3
```

# DATA QUERY LANGUAGE (DQL):

## 1) {Select Command}:-

### A) To Display All records in a Table :-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for Management, Instance, Performance, Administration, and Schemas. The main area has a 'Query 1' tab with the SQL command: '1 • Select \* from Customers;'. Below it is a 'Result Grid' showing 10 rows of customer data:

customer_id	customer_name	email	address	area	pincode	registration_date	customer_segment	total_orders	avg_order_valu
2	Priya Verma	priya.verma@example.com	456 Sector 21, Delhi	Dwarka	110075	2022-12-20	Regular	10	450.00
3	Amit Gupta	amit.gupta@example.com	789 MG Road, Bangalore	Koramangala	560034	2023-02-10	Premium	18	720.75
4	Sneha Kapoor	sneha.kapoor@example.com	159 Park Street, Kolkata	Salt Lake	700091	2023-03-05	Premium	8	320.00
5	Vikram Singh	vikram.singh@example.com	357 Lajpat Nagar, Delhi	Lajpat Nagar	110024	2022-11-25	Premium	22	890.90
6	Neha Mehta	neha.mehta@example.com	246 Banjara Hills, Hyderabad	Banjara Hills	500034	2023-04-12	Regular	12	480.20
7	Arjun Mahotra	arjun.mahotra@example.com	951 Shivaji Park, Pune	Shivaji Park	411002	2023-05-08	Premium	25	940.00
8	Divya Nair	divya.nair@example.com	753 MG Road, Chennai	T. Nagar	600017	2023-06-14	Regular	9	350.75
9	Rohit Chawla	rohit.chawla@example.com	852 Sadar Bazar, Gurgaon	Sadar Bazar	122001	2023-07-10	Premium	30	1020.50
...	...	...	...	...	...	...	...	...	...

The bottom output pane shows the executed query: '1 21:51:36 Select \* from Customers LIMIT 0, 1000' and a message: '20 row(s) returned'.

### B) {Select} command with where clause:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for Management, Instance, Performance, Administration, and Schemas. The main area has a 'Query 1' tab with the SQL command: '1 SELECT customer\_name, total\_orders FROM Customers WHERE total\_orders > 10;'. Below it is a 'Result Grid' showing 10 rows of filtered customer data:

customer_name	total_orders
Amit Gupta	18
Vikram Singh	22
Neha Mehta	12
Arjun Mahotra	25
Rohit Chawla	30
Sandeep Rao	27
Meera Shah	11
Kunal Bansal	20
Tarun Agarwal	14
...	...

The bottom output pane shows the executed query: '1 22:00:15 SELECT customer\_name, total\_orders FROM Customers WHERE total\_orders > 10 LIMIT 0, 1000' and a message: '13 row(s) returned'.

## 2) {Show command}:-

### A) To display all tables in a database:-

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Local instance MySQL80.
- Query Editor:** Query 1, SQL File 2\*, containing the command: `1 • Show Tables;`
- Result Grid:** Displays the results of the query, showing the tables in the current database: `Tables_in_blinkit`, which includes `customers`, `deliverperformance`, `inventory`, `orderdetails`, `orders`, and `products`.
- Output:** Action Output shows the query was run at 22:07:28, returning 6 row(s).

### B) To Display database:-

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Local instance MySQL80.
- Query Editor:** Query 1, SQL File 2\*, containing the command: `1 show databases;`
- Result Grid:** Displays the results of the query, showing the databases: `Database`, which includes `blinkit`, `information_schema`, `mysql`, `performance_schema`, `pizza_sales_analysis`, and `sys`.
- Output:** Action Output shows the query was run at 22:11:23, returning 6 row(s).

### 3) Like QUERY :

A) Using '%':-

The screenshot shows the MySQL Workbench interface. The left sidebar has sections for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area shows a query editor with the following SQL code:

```
1 SELECT customer_id, customer_name, email
2 FROM Customers
3 WHERE customer_name LIKE '%R%';
```

The result grid displays three rows of data from the Customers table:

customer_id	customer_name	email
9	Rohit Chawla	rohit.chawla@example.com
14	Ritika Roy	ritika.roy@example.com
17	Rajesh Kumar	rajeesh.kumar@example.com

The output pane shows the query and its execution message:

```
1 22:20:00 SELECT customer_id, customer_name, email FROM Customers WHERE customer_name LIKE '%R%' LIMIT 0, 1000 3 row(s) returned
```

B) Using '\_':-

The screenshot shows the MySQL Workbench interface. The left sidebar has sections for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area shows a query editor with the following SQL code:

```
1 • SELECT customer_id, customer_name
2   FROM Customers
3   WHERE customer_name LIKE '_a%';
```

The result grid displays six rows of data from the Customers table:

customer_id	customer_name
11	Sandeep Rao
15	Tarun Aggarwal
17	Rajesh Kumar
19	Yashwant Deshmukh
20	Kavita Saxena
21	Harish Bhatt

The output pane shows the query and its execution message:

```
1 22:24:15 SELECT customer_id, customer_name FROM Customers WHERE customer_name LIKE '_a%' LIMIT 0, 1000 6 row(s) returned
```

#### 4) Limit Clause:-

The screenshot shows the MySQL Workbench interface. In the top query editor, the following SQL code is written:

```
1 • SELECT * FROM Customers
2   LIMIT 5;
```

The result grid displays the first 5 rows of the 'Customers' table:

customer_id	customer_name	email	address	area	pincode	registration_date	customer_segment	total_orders	avg_order_value
2	Priya Verma	priya.verma@example.com	456 Sector 21, Delhi	Dwarka	110075	2022-12-20	Regular	10	450.00
3	Amit Gupta	amit.gupta@example.com	789 MG Road, Bangalore	Koramangala	560034	2023-02-10	Premium	18	720.75
4	Sneha Kapoor	sneha.kapoor@example.com	159 Park Street, Kolkata	Salt Lake	700091	2023-03-05	Regular	8	320.00
5	Vikram Singh	vikram.singh@example.com	357 Lajpat Nagar, Delhi	Lajpat Nagar	110024	2022-11-25	Premium	22	890.90
6	Neha Mehta	neha.mehta@example.com	246 Banjara Hills, Hyderabad	Banjara Hills	500034	2023-04-12	Regular	12	480.20

The output pane shows the executed query and its results:

```
1 22:44:49 SELECT * FROM Customers LIMIT 5
Message
5 row(s) returned
```

#### 5) Group by clause:-

##### A) Group by:-

The screenshot shows the MySQL Workbench interface. In the top query editor, the following SQL code is written:

```
1 • SELECT area, COUNT(customer_id) AS total_customers
2   FROM Customers
3   GROUP BY area;
```

The result grid displays the count of customers for each area:

area	total_customers
Dwarka	1
Koramangala	1
Salt Lake	1
Lajpat Nagar	1
Banjara Hills	1
Shivaji Park	1

The output pane shows the executed query and its results:

```
1 22:47:59 SELECT area, COUNT(customer_id) AS total_customers FROM Customers GROUP BY area LIMIT 0, 1000
Message
20 row(s) returned
```

## B) Group By with Having clause:-

The screenshot shows the MySQL Workbench interface with a query editor and results grid.

**Query Editor:**

```
1 • SELECT category, AVG(price) AS avg_price
2   FROM Products
3   GROUP BY category
4   HAVING avg_price > 200;
```

**Results Grid:**

category	avg_price
Dry Fruits	500
Groceries	325
Cooking Essentials	900
Personal Care	220

**Output Panel:**

```
1 23:00:15 SELECT category, AVG(price) AS avg_price FROM Products GROUP BY category HAVING avg_price > 200 L... 4 row(s) returned
```

## 6) Order By clause:-

### A) Order by ASC:-

The screenshot shows the MySQL Workbench interface with a query editor and results grid.

**Query Editor:**

```
1 • SELECT *
2   FROM Customers
3   ORDER BY registration_date ASC;
```

**Results Grid:**

customer_name	email	address	area	pincode	registration_date	customer_segment	total_orders	avg_order_value	gender
Vikram Singh	vikram.singh@example.com	357 Lajpat Nagar, Delhi	Lajpat Nagar	110024	2022-11-25	Premium	22	890.90	Male
Priya Verma	priya.verma@example.com	456 Sector 21, Delhi	Dwarka	110075	2022-12-20	Regular	10	450.00	Female
Amit Gupta	amit.gupta@example.com	789 MG Road, Bangalore	Koramangala	560034	2023-02-10	Premium	18	720.75	Male
Sneha Kapoor	sneha.kapoor@example.com	159 Park Street, Kolkata	Salt Lake	700091	2023-03-05	Regular	8	320.00	Female
Neha Mehta	neha.mehta@example.com	246 Banjara Hills, Hyderabad	Banjara Hills	500034	2023-04-12	Regular	12	480.20	Female

**Output Panel:**

```
1 23:05:38 SELECT * FROM Customers ORDER BY registration_date ASC LIMIT 0, 1000
20 row(s) returned
```

## B) Order by DESC:-

The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```
1 • SELECT *
2   FROM Products
3   ORDER BY price DESC;
```

The results grid displays the following data:

product_id	product_name	category	brand	price	mrp	margin_percentage	min_stock_level	max_stock_level
3	Cold Pressed Olive Oil	Cooking Essentials	Figaro	900	950.00	5.00	5	40
2	Basmati Rice 5kg	Groceries	Daawat	600	650.00	8.00	10	100
1	Organic Almonds	Dry Fruits	NutriFresh	500	550.00	10.00	5	50
11	Shampoo 500ml	Personal Care	Head & Shoulders	350	400.00	12.00	10	50
15	Body Lotion 300ml	Personal Care	Nivea	250	280.00	10.00	5	30
8	Detergent Powder 2kg	Household	SurfExcel	220	250.00	12.00	10	80

The output pane shows the query and its execution details:

```
1 23:12:43 SELECT * FROM Products ORDER BY price DESC LIMIT 0, 1000
21 row(s) returned
```

## 7) Built in Function:-

### A) Concat:-

The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```
1 • SELECT customer_id, CONCAT(customer_name, ' - ', customer_segment) AS customer_details
2   FROM Customers;
```

The results grid displays the following data:

customer_id	customer_details
2	Priya Verma - Regular
3	Amit Gupta - Premium
4	Sneha Kapoor - Regular
5	Vikram Singh - Premium
6	Neha Mehta - Regular
7	Arjun Malhotra - Premium

The output pane shows the query and its execution details:

```
1 23:23:25 SELECT customer_id,CONCAT(customer_name,'-',customer_segment) AS customer_details FROM Customers... 20 row(s) returned
```

## B) Lower:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for Management, Instance, and Performance. The main area has a query editor titled "Query 1" with the following SQL code:

```
1 •  SELECT product_id, LOWER(product_name) AS lowercase_product_name
2   FROM Products;
```

The results are displayed in a grid:

product_id	lowercase_product_name
1	organic almonds
2	basmati rice 5kg
3	cold pressed olive oil
4	whole wheat bread
5	dark chocolate 70%
6	coca cola Zi

The output pane shows the execution log:

#	Time	Action	Message
1	23:30:33	SELECT product_id, LOWER(product_name) AS lowercase_product_name FROM Products LIMIT 0, 1000	21 row(s) returned

## C) Upper:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for Management, Instance, and Performance. The main area has a query editor titled "Query 1" with the following SQL code:

```
1 •  SELECT customer_id, UPPER(customer_name) AS uppercase_name
2   FROM Customers;
```

The results are displayed in a grid:

customer_id	uppercase_name
2	PRIYA VERMA
3	AMIT GUPTA
4	SNEHA KAPOOR
5	VIKRAM SINGH
6	NEHA MEHTA
7	ARJUN MALHOTRA

The output pane shows the execution log:

#	Time	Action	Message
1	23:33:27	SELECT customer_id, UPPER(customer_name) AS uppercase_name FROM Customers LIMIT 0, 1000	20 row(s) returned

#### D) Replace:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has two tabs: 'Query 1' and 'SQL File 2'. The 'Query 1' tab contains the following SQL code:

```
1 • SELECT product_id, REPLACE(brand, 'Amul', 'Nestle') AS updated_brand
2   FROM Products;
```

The results grid shows the output of the query:

product_id	updated_brand
1	NutriFresh
2	Daawat
3	Figaro
4	Harvest Gold
5	Nestle
6	Coca Cola

The output pane at the bottom shows the query and its execution details:

#	Time	Action
1	23:43:01	SELECT product_id, REPLACE(brand, 'Amul', 'Nestle') AS updated_brand FROM Products LIMIT 0, 1000

Message: 21 row(s) returned

#### E) Reverse:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has two tabs: 'Query 1' and 'SQL File 2'. The 'Query 1' tab contains the following SQL code:

```
1 • SELECT product_id, product_name, REVERSE(product_name) AS reversed_name
2   FROM Products;
```

The results grid shows the output of the query:

product_id	product_name	reversed_name
1	Organic Almonds	sdnhomA cinagrO
2	Basmat Rice 5kg	gk5 ecR itamsaB
3	Cold Pressed Olive Oil	lO evlO desserP dloC
4	Whole Wheat Bread	daerB taehW eloHW
5	Dark Chocolate 70%	%07 etalocoC kraD
6	Coca Cola 2L	L2 aloC acoC

The output pane at the bottom shows the query and its execution details:

#	Time	Action
1	23:46:10	SELECT product_id, product_name, REVERSE(product_name) AS reversed_name FROM Products LIMIT 0, 10...

Message: 21 row(s) returned

## F) Length:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for Management, Instance, Performance, Administration, and Schemas. The main area has two tabs: 'Query 1' and 'SQL File 2'. The 'Query 1' tab contains the following SQL code:

```
1 • SELECT customer_id, customer_name, LENGTH(customer_name) AS name_length
2 FROM Customers;
```

The results are displayed in a 'Result Grid' table:

customer_id	customer_name	name_length
2	Priya Verma	11
3	Amit Gupta	10
4	Sneha Kapoor	12
5	Vikram Singh	12
6	Neha Mehta	10
7	Arjun Malhotra	14

The output pane shows the execution message:

```
1 23:50:55 SELECT customer_id, customer_name, LENGTH(customer_name) AS name_length FROM Customers LIMIT 0, ... 20 row(s) returned
```

## G) Substring:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for Management, Instance, Performance, Administration, and Schemas. The main area has two tabs: 'Query 1' and 'SQL File 2'. The 'Query 1' tab contains the following SQL code:

```
1 • SELECT order_id, SUBSTRING(order_date, 1, 4) AS order_year FROM Orders;
2
3
```

The results are displayed in a 'Result Grid' table:

order_id	order_year
2	2024
3	2024
4	2024
5	2024
6	2024
7	2024

The output pane shows the execution message:

```
1 00:56:17 SELECT order_id, SUBSTRING(order_date, 1, 4) AS order_year FROM Orders LIMIT 0, 1000
20 row(s) returned
```

## 8) Math Function

### A) Absolute:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has a 'Query 1' tab with the following SQL code:

```
1 •  SELECT product_id, price, mrp, ABS(mrp - price) AS discount_amount FROM Products;
```

The results are displayed in a grid:

product_id	price	mrp	discount_amount
1	500	550.00	50
2	600	650.00	50
3	900	950.00	50
4	50	60.00	10
5	150	180.00	30
6	90	100.00	10

The output pane shows the message: "21 row(s) returned".

### B) Mod():-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has a 'Query 1' tab with the following SQL code:

```
1 •  select mod(15,5) as remainders;
```

The results are displayed in a grid:

remainder
0

The output pane shows the message: "1 row(s) returned".

### C) Floor():-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, PERFORMANCE, and INFORMATION. The main area has a query editor titled 'Query 1' with the SQL command: `SELECT FLOOR(7.8) AS floored_value;`. Below the query is a result grid showing one row with the value 7. The output pane at the bottom shows the execution log: '1 01:07:53 SELECT FLOOR(7.8) AS floored\_value LIMIT 0, 1000' and '1 row(s) returned'.

### D) Ceiling():-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, PERFORMANCE, and INFORMATION. The main area has a query editor titled 'Query 1' with the SQL command: `SELECT CEILING(7.2) AS ceiling_value;`. Below the query is a result grid showing one row with the value 8. The output pane at the bottom shows the execution log: '1 01:11:24 SELECT CEILING(7.2) AS ceiling\_value LIMIT 0, 1000' and '1 row(s) returned'.

## E) Exponential():-

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Query' tab, there is a 'Result Grid' section. It displays the output of the following SQL query:

```
1 • SELECT EXP(2) AS exponential_value;
```

The result grid shows one row with the column 'exponential\_value' containing the value '7.38905609893065'. Below the result grid, the 'Output' pane shows the log entry for this query:

#	Time	Action	Message
1	01:14:39	SELECT EXP(2) AS exponential_value LIMIT 0, 1000	1 row(s) returned

## F) Power():-

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Query' tab, there is a 'Result Grid' section. It displays the output of the following SQL query:

```
1 • SELECT POWER(3, 4) AS result;
```

The result grid shows one row with the column 'result' containing the value '81'. Below the result grid, the 'Output' pane shows the log entry for this query:

#	Time	Action	Message
1	01:16:44	SELECT POWER(3, 4) AS result LIMIT 0, 1000	1 row(s) returned

## G) Square root():-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has a 'Query 1' tab with the SQL command: `SELECT SQRT(25) AS result;`. The results grid shows one row with the value 5. Below the results, the 'Output' pane displays the query log: `1 01:32:44 SELECT SQRT(25) AS result LIMIT 0, 1000`.

## 9) Aggregate function:-

### A) Count:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has a 'Query 1' tab with the SQL command: `SELECT COUNT(*) AS total_orders FROM Orders;`. The results grid shows one row with the value 20. Below the results, the 'Output' pane displays the query log: `1 01:37:32 SELECT COUNT(*) AS total_orders FROM Orders LIMIT 0, 1000`.

## B) Average:-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has a query editor titled "Query 1" with the SQL command:

```
1 • SELECT AVG(price) AS avg_price FROM Products;
```

The result grid shows one row with the column "avg\_price" containing the value "200.95238095238096". Below the result grid, the message "1 row(s) returned" is displayed.

## C) Sum():-

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, and PERFORMANCE. The main area has a query editor titled "Query 1" with the SQL command:

```
1 • SELECT SUM(total_amount) AS total_revenue FROM Orders;
```

The result grid shows one row with the column "total\_revenue" containing the value "12605.15". Below the result grid, the message "1 row(s) returned" is displayed.

## 10) Date () function :-

### A) Curdate():-

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Information' section, there is a note: 'No object selected'. The main area has two tabs: 'Query 1' and 'SQL File 2'. The 'Query 1' tab contains the SQL command:

```
1 • SELECT CURRENT_DATE() AS today_date;
```

The 'Result Grid' shows the output:

today_date
2025-03-08

The 'Output' tab shows the execution details:

#	Time	Action
1	01:49:33	SELECT CURRENT_DATE() AS today_date LIMIT 0, 1000

Message: 1 row(s) returned.

### B) Now():-

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Information' section, there is a note: 'No object selected'. The main area has two tabs: 'Query 1' and 'SQL File 2'. The 'Query 1' tab contains the SQL command:

```
1 • SELECT NOW() AS current_datetime;
```

The 'Result Grid' shows the output:

current_datetime
2025-03-08 01:52:55

The 'Output' tab shows the execution details:

#	Time	Action
1	01:52:55	SELECT NOW() AS current_datetime LIMIT 0, 1000

Message: 1 row(s) returned.

### C) Last\_Day:-

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Information' section, there is a note: 'No object selected'. The main area has two tabs: 'Query 1' and 'SQL File 2'. The 'Query 1' tab contains the SQL command:

```
1 • SELECT LAST_DAY(NOW()) AS last_day_of_month;
```

The 'Result Grid' shows the output:

last_day_of_month
2025-03-31

The 'Output' tab shows the execution details:

#	Time	Action
1	01:57:52	SELECT LAST_DAY(NOW()) AS last_day_of_month LIMIT 0, 1000

Message: 1 row(s) returned.

#### D) Date\_Format():-

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows various management and performance metrics.
- Query Editor:** Title: "Query 1 - SQL File 2".  
SQL Query:

```
1 • SELECT DATE_FORMAT(NOW(), "%Y-%m-%d") AS formatted_date;
```
- Result Grid:** Shows the output of the query: "formatted\_date" with value "2025-03-08".
- Action Output:** Shows the execution log: "1 02:01:36 SELECT DATE\_FORMAT(NOW(), "%Y-%m-%d") AS formatted\_date LIMIT 0, 1000".
- Output:** Message: "1 row(s) returned".

#### E) Datediff():-

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows various management and performance metrics.
- Query Editor:** Title: "Query 1 - SQL File 2".  
SQL Query:

```
1 • SELECT DATEDIFF('2025-03-10', '2025-03-01') AS days_difference;
```
- Result Grid:** Shows the output of the query: "days\_difference" with value "9".
- Action Output:** Shows the execution log: "1 02:05:57 SELECT DATEDIFF('2025-03-10', '2025-03-01') AS days\_difference LIMIT 0, 1000".
- Output:** Message: "1 row(s) returned".

#### F) Union():-

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows various management and performance metrics.
- Query Editor:** Title: "Query 1 - SQL File 2".  
SQL Query:

```
1 • SELECT email AS details FROM Customers  
2 UNION  
3 SELECT payment_method FROM Orders;
```
- Result Grid:** Shows the output of the query: "details" with values:
  - priya.verma@example.com
  - amit.gupta@example.com
  - sneha.kapoor@example.com
  - vikram.singh@example.com
  - neha.mehta@example.com
  - arjun.mahotra@example.com
- Action Output:** Shows the execution log: "1 02:11:15 SELECT email AS details FROM Customers UNION SELECT payment\_method FROM Orders".
- Output:** Message: "25 row(s) returned".

## 11) SUB-QUERY:

### A) SINGLE ROW SUBQUERY():-

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:`1 • SELECT * FROM Customers  
2 WHERE total_orders = (SELECT MAX(total_orders) FROM Customers);  
3  
4  
5`

The result grid displays one row of data from the "Customers" table:

customer_id	customer_name	email	address	area	pincode	registration_date	customer_segment	total_orders	avg_order_value	gen	last_order
9	Rohit Chawla	rohit.chawla@example.com	852 Sadar Bazar, Gurgaon	Sadar Bazar	122001	2023-07-10	Premium	30	1020.50	Male	2023-07-10

The output pane shows the message: "1 02:17:08 SELECT \* FROM Customers WHERE total\_orders = (SELECT MAX(total\_orders) FROM Customers) LIMIT 0, 1... 1 row(s) returned".

### B) Multiple row Subquery:-

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:`1 • SELECT product_name, category, price  
2 FROM Products  
3 WHERE price IN (SELECT price FROM Products WHERE category = 'Beverages');  
4  
5`

The result grid displays three rows of data from the "Products" table:

product_name	category	price
Coca Cola 2L	Beverages	90
Green Tea 250g	Beverages	200
Green Grapes 500g	Fruits	90

The output pane shows the message: "1 02:21:13 SELECT product\_name, category, price FROM Products WHERE price IN (SELECT price FROM Products WHERE category = 'Beverages') 3 row(s) returned".

### C) Multiple Column Subquery:-

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:`1 • SELECT product_id, date, stock_received, damaged_stock  
2 FROM Inventory  
3 WHERE (product_id, stock_received) IN  
4 (SELECT product_id, max_stock_level  
5 FROM Products);`

The result grid displays two rows of data from the "Inventory" table:

product_id	date	stock_received	damaged_stock
1	2024-03-01	50	2
3	2024-03-03	40	3

The output pane shows the message: "1 02:40:43 SELECT product\_id, date, stock\_received, damaged\_stock FROM Inventory WHERE (product\_id, stock\_received) IN (SELECT product\_id, max\_stock\_level FROM Products) 2 row(s) returned".

## D) Multiple Table Query:-

```

1 • SELECT
2     customer_id,
3     (SELECT customer_name FROM Customers WHERE Customers.customer_id = C.customer_id) AS customer_name,
4     (SELECT email FROM Customers WHERE Customers.customer_id = C.customer_id) AS email,
5     (SELECT product_id FROM Products ORDER BY product_id LIMIT 1) AS product_id,
6     (SELECT product_name FROM Products WHERE Products.product_id =
7         (SELECT product_id FROM Products ORDER BY product_id LIMIT 1)) AS product_name,
8     (SELECT category FROM Products WHERE Products.product_id =
9         (SELECT product_id FROM Products ORDER BY product_id LIMIT 1)) AS category,
10    (SELECT price FROM Products WHERE Products.product_id =
11        (SELECT product_id FROM Products ORDER BY product_id LIMIT 1)) AS price

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

customer_id	customer_name	email	product_id	product_name	category	price
2	Priya Verma	priya.verma@example.com	1	Organic Almonds	Dry Fruits	500
3	Amit Gupta	amit.gupta@example.com	1	Organic Almonds	Dry Fruits	500
4	Sneha Kapoor	sneha.kapoor@example.com	1	Organic Almonds	Dry Fruits	500
5	Vikram Singh	vikram.singh@example.com	1	Organic Almonds	Dry Fruits	500
6	Neha Mehta	neha.mehta@example.com	1	Organic Almonds	Dry Fruits	500
7	Ariun Malhotra	ariun.malhotra@example.com	1	Organic Almonds	Dry Fruits	500

No object selected

Output:

#	Time	Action	Message
1	02:52:00	SELECT customer_id, (SELECT customer_name FROM Customers WHERE Customers.customer_id = C.cus...	20 row(s) returned

## 12) Joins

### A) Inner join:-

```

1 • SELECT
2     c.customer_id,
3     c.customer_name,
4     c.email,
5     p.product_id,
6     p.product_name,
7     p.category,
8     p.price,
9     i.stock_received,
10    i.damaged_stock
11   FROM Customers c
12   INNER JOIN Orders o ON c.customer_id = o.customer_id
13   INNER JOIN OrderDetails od ON o.order_id = od.order_id
14   INNER JOIN Products p ON od.product_id = p.product_id
15   INNER JOIN Inventory i ON p.product_id = i.product_id

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

customer_id	customer_name	email	product_id	product_name	category	price	stock_received	damaged_stock

No object selected

Result 3 | Output:

#	Time	Action	Message
1	13:39:33	SELECT c.customer_id, c.customer_name, c.email, p.product_id, p.product_name, p.category...	0 row(s) returned

## B) Left Join:-

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```

1 • SELECT
2     c.customer_id,
3     c.customer_name,
4     c.email,
5     o.order_id,
6     o.order_date,
7     o.total_amount
8 FROM Customers c
9 LEFT JOIN Orders o ON c.customer_id = o.customer_id;
10
11
12
  
```

The result grid displays the following data:

customer_id	customer_name	email	order_id	order_date	total_amount
2	Priya Verma	priya.verma@example.com	2	2024-03-02 14:15:00	320.50
3	Amit Gupta	amit.gupta@example.com	3	2024-03-03 09:45:00	785.00
4	Sneha Kapoor	sneha.kapoor@example.com	4	2024-03-04 12:20:00	650.75
5	Vikram Singh	vikram.singh@example.com	5	2024-03-05 18:10:00	920.00
6	Neha Mehta	neha.mehta@example.com	6	2024-03-06 08:00:00	410.25

The output pane shows the following log entry:

```

#   Time      Action
1 13:52:22  SELECT  c.customer_id, c.customer_name, c.email, o.order_id, o.order_date, o.total_amount ... 20 row(s) returned
  
```

## C) Right Join:-

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```

1 • SELECT
2     c.customer_id,
3     c.customer_name,
4     o.order_id,
5     o.order_date,
6     p.product_id,
7     p.product_name,
8     p.category
9 FROM Orders o
10 RIGHT JOIN Customers c ON o.customer_id = c.customer_id
11 RIGHT JOIN Orderdetails od ON o.order_id = od.order_id
12 RIGHT JOIN Products p ON od.product_id = p.product_id
  
```

The result grid displays the following data:

customer_id	customer_name	order_id	order_date	product_id	product_name	category
HULL	HULL	HULL	HULL	1	Organic Almonds	Dry Fruits
HULL	HULL	HULL	HULL	2	Basmati Rice 5kg	Groceries
HULL	HULL	HULL	HULL	3	Cold Pressed Olive Oil	Cooking Essentials
HULL	HULL	HULL	HULL	4	Whole Wheat Bread	Bakery
HULL	HULL	HULL	HULL	5	Dark Chocolate 70%	Snacks

The output pane shows the following log entry:

```

#   Time      Action
1 14:02:07  SELECT  c.customer_id, c.customer_name, o.order_id, o.order_date, p.product_id, p.product_name, p.category ...
  
```

## D) Full join:-

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```

1 • SELECT
2     c.customer_id,
3     c.customer_name,
4     o.order_id,
5     o.order_date,
6     o.total_amount
7   FROM Customers c
8   LEFT JOIN Orders o ON c.customer_id = o.customer_id
9
10 UNION
11
12 SELECT
13     c.customer_id,
14     c.customer_name,
15     o.order_id,
16     o.order_date,
17     o.total_amount
  
```

The result grid displays the following data:

customer_id	customer_name	order_id	order_date	total_amount
2	Priya Verma	2	2024-03-02 14:15:00	320.50
3	Ankit Gupta	3	2024-03-03 09:45:00	785.00
4	Sneha Kapoor	4	2024-03-04 12:20:00	650.75
5	Vikram Singh	5	2024-03-05 18:10:00	920.00
6	Neha Mehta	6	2024-03-06 08:00:00	410.25

The output pane shows the message: "20 row(s) returned".

## E) Views:-

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```

1 • CREATE VIEW AvailableProducts AS
2
3     SELECT
4         p.product_id,
5         p.product_name,
6         p.category,
7         p.brand,
8         p.price,
9         i.stock_received - i.damaged_stock AS available_stock
10    FROM Products p
11   INNER JOIN Inventory i ON p.product_id = i.product_id;
  
```

The output pane shows the message: "0 row(s) affected".

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```

1 • SELECT * FROM AvailableProducts;
2
  
```

The result grid displays the following data:

product_id	product_name	category	brand	price	available_stock
1	Organic Almonds	Dry Fruits	NutriFresh	500	48
2	Basmati Rice 5kg	Groceries	Daawat	600	29
3	Cold Pressed Olive Oil	Cooking Essentials	Figaro	900	37
4	Whole Wheat Bread	Bakery	Harvest Gold	50	25
5	Dark Chocolate 70%	Snacks	Amul	150	55

The output pane shows the message: "21 row(s) returned".

## Conclusion

The project successfully demonstrated how **SQL can be leveraged to manage, analyze, and optimize business operations** in an e-commerce environment like Blinkit. The insights gained from this analysis can help improve **customer engagement, stock management, and delivery efficiency**, ultimately leading to better business decisions.

The SQL project on the **Blinkit dataset** provided valuable insights into customer behavior, order patterns, product inventory, and delivery performance. By designing and querying relational tables efficiently, we were able to analyze key aspects of the business operations.