

Group A: Study of Databases

2. Install and configure client and server of MySQL. (Show all commands and necessary steps for installation and configuration)

STEP 1:

The screenshot shows a Yahoo India search results page for the query "mysql". The top result is the official MySQL website. The page includes a sidebar with links for Downloads, Partners, Customers, Why MySQL, News & Events, and Services. To the right, there's a large MySQL logo and a brief description of what MySQL is.

STEP 2:

The screenshot shows the MySQL Downloads page. It features a prominent section for MySQL HeatWave, described as "One MySQL Database service for OLTP, OLAP, and ML". Below this, there are sections for "HeatWave AutoML", "Faster Performance" (with a comparison to Amazon RDS, Aurora, Redshift, and Snowflake), and "Lower Total Cost of Ownership" (with a comparison to Amazon RDS, Aurora, Redshift, and Snowflake). At the bottom, there are links for "Try Free", "Technical Guides", and newsletter subscriptions.

STEP 3:

The screenshot shows a web browser window with the URL <https://dev.mysql.com/downloads/>. The main content is the 'MySQL Cluster CGE' page. It features a sidebar with links like 'Sector Improve and Expand Services While Cutting Costs On Demand', 'Beginner's Guide to Implementing InnoDB ClusterSet with MySQL Shell On Demand', and '20x Faster Analytics, 25x Faster Machine Learning with MySQL HeatWave on AWS On Demand'. The main area describes MySQL Cluster as a real-time open source transactional database and lists download options: 'MySQL Cluster', 'MySQL Cluster Manager', and 'Plus, everything in MySQL Enterprise Edition'. Below this are 'Learn More', 'Customer Download' (with a note about Patches & Updates Tab and Product Search), and 'Trial Download'. A link to 'MySQL Community (GPL) Downloads' is also present. The bottom of the page includes contact information for sales and online contact, along with a search bar and system status indicators.

STEP 4:

The screenshot shows a web browser window with the URL <https://dev.mysql.com/downloads/file/?id=516926>. The main content is the 'MySQL Community Downloads' page. It features a call-to-action for logging in or signing up for a free account. It lists advantages of an Oracle Web Account, such as fast access to MySQL software downloads, download of technical White Papers and Presentations, posting messages in the MySQL Discussion Forums, and reporting bugs in the MySQL bug system. Below this is a section for Oracle SSO authentication with 'Login » using my Oracle Web account' and 'Sign Up » for an Oracle Web account'. A note explains that MySQL.com uses Oracle SSO for authentication. At the bottom, there is a link to start the download without creating an account.

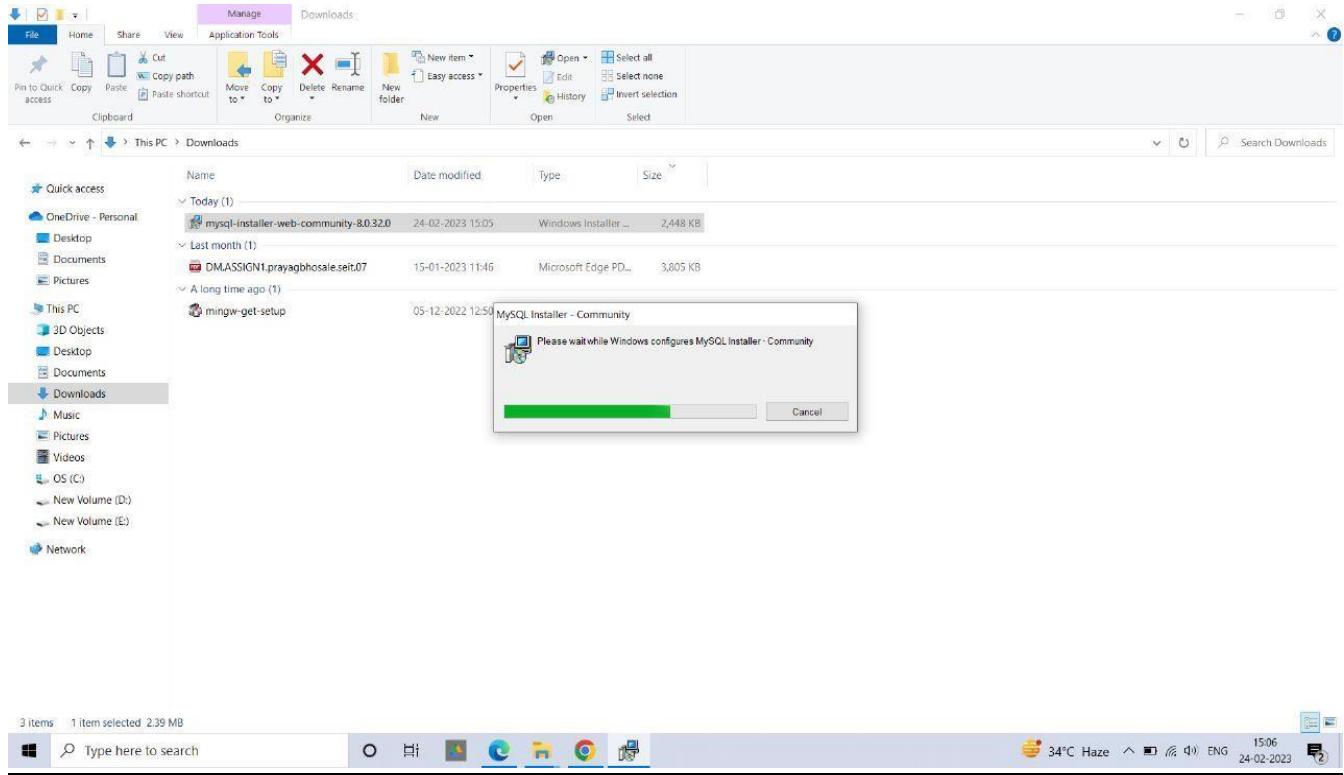
[No thanks, just start my download.](#)

ORACLE © 2023 Oracle

[Privacy / Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Cookie Preferences](#)

The screenshot shows a web browser window with the URL <https://dev.mysql.com/get/Downloads/MySQL/installer/mysql-installer-web-community-8.0.32.0.msi>. The main content is the MySQL Installer download page. It shows the file name 'mysql-installer-web-community-8.0.32.0.msi' and its size '1.1 GB'. Below this is a large red button labeled 'Download MySQL Installer'. The bottom of the page includes a search bar and system status indicators.

STEP 5:



STEP 6:

A screenshot of a web browser displaying the MySQL Community Downloads page. The URL is 'dev.mysql.com/downloads/installer/'. The page title is 'MySQL Community Downloads' under 'MySQL Installer'. The 'General Availability (GA) Releases' tab is selected. Under 'MySQL Installer 8.0.32', there are two download options for 'Windows (x86, 32-bit), MSI Installer': one for version 8.0.32 (2.4M) and another for version 8.0.32 (437.3M). Both have 'Download' buttons. A note at the bottom suggests using MD5 checksums and GnuPG signatures for verification. The footer includes the Oracle logo and copyright information.

MySQL - Yahoo India Search Results | MySQL :: Download MySQL Installer

Type here to search

MySQL Community Downloads

MySQL Installer

General Availability (GA) Releases Archives

MySQL Installer 8.0.32

Select Operating System: Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer 8.0.32 2.4M Download
(mysql-installer-web-community-8.0.32.0.msi)
MD5: 0f082590f8338adc614e9dc5cb80ca6b | Signature

Windows (x86, 32-bit), MSI Installer 8.0.32 437.3M Download
(mysql-installer-community-8.0.32.0.msi)
MD5: a29b5817cba2c7bc0e0b97e897c2591f | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

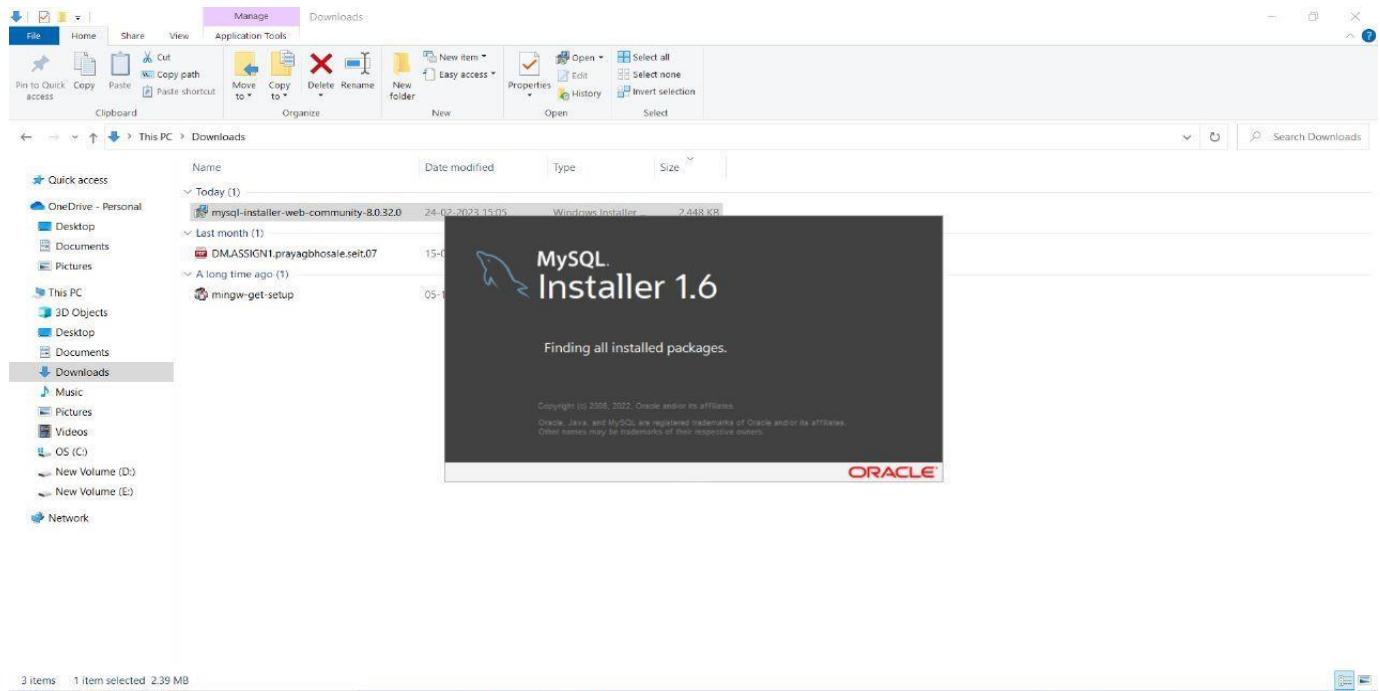
ORACLE © 2023 Oracle

https://dev.mysql.com/downloads/file/?id=516926

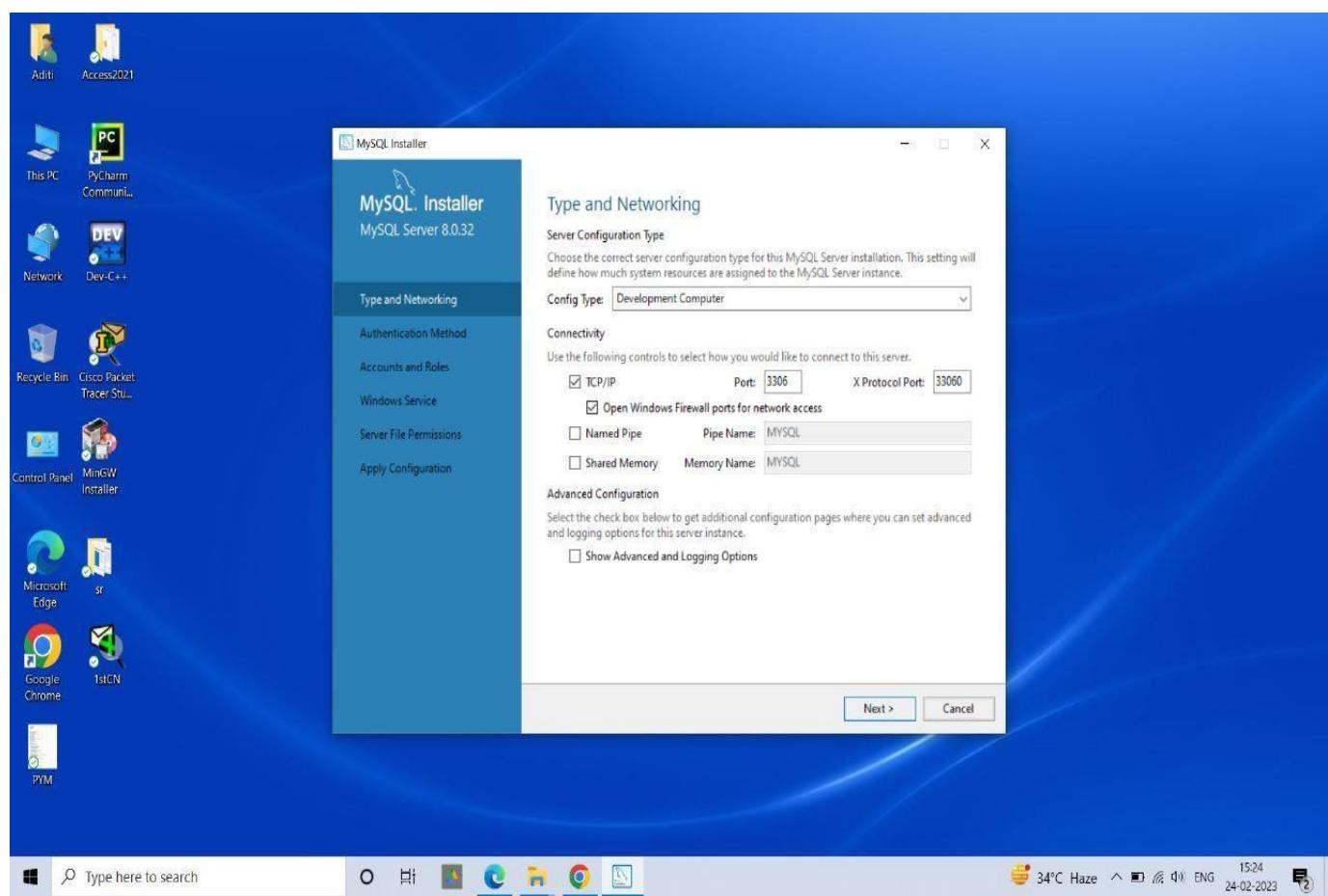
Type here to search

34°C Haze 15:04 ENG 24-02-2023

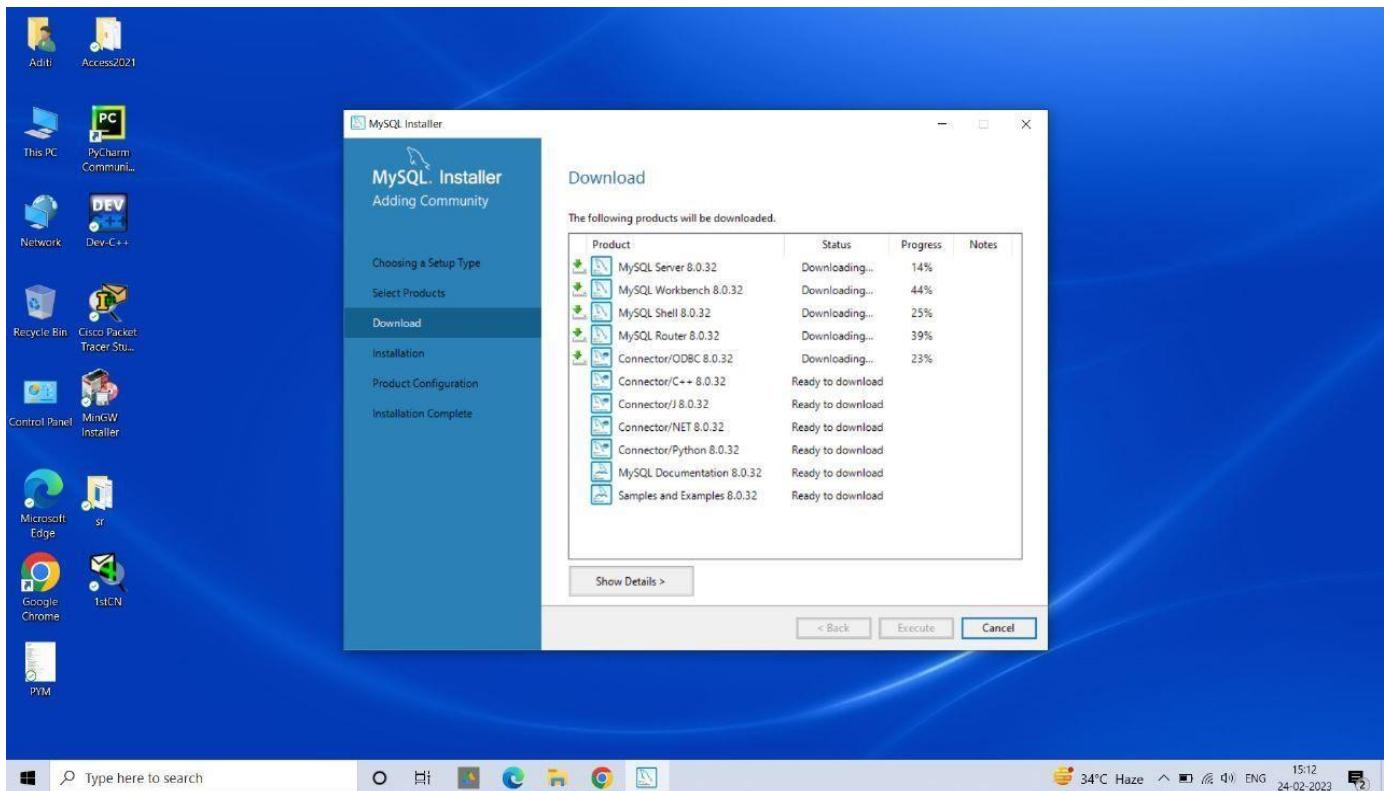
STEP 7:



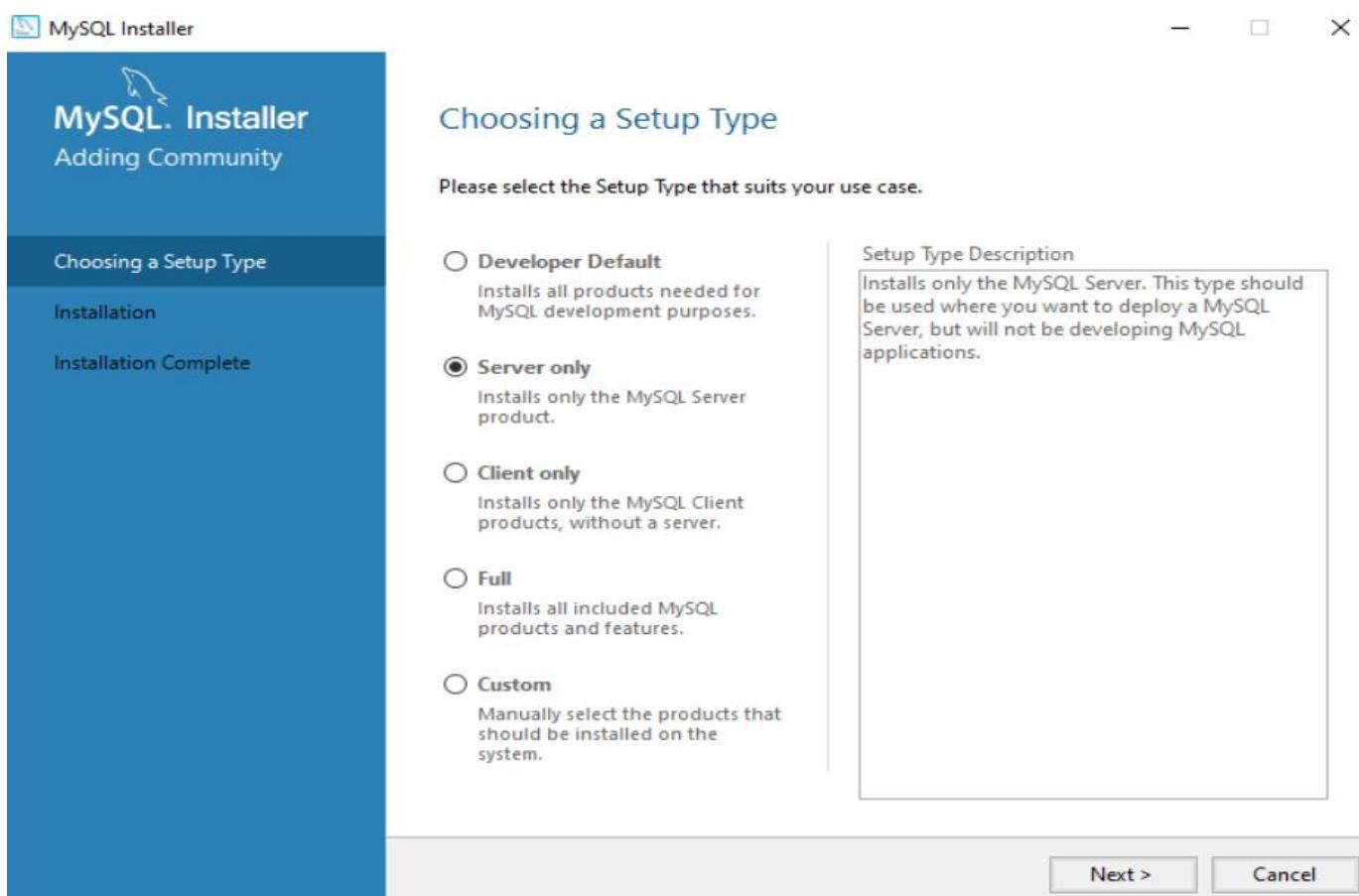
STEP 8:



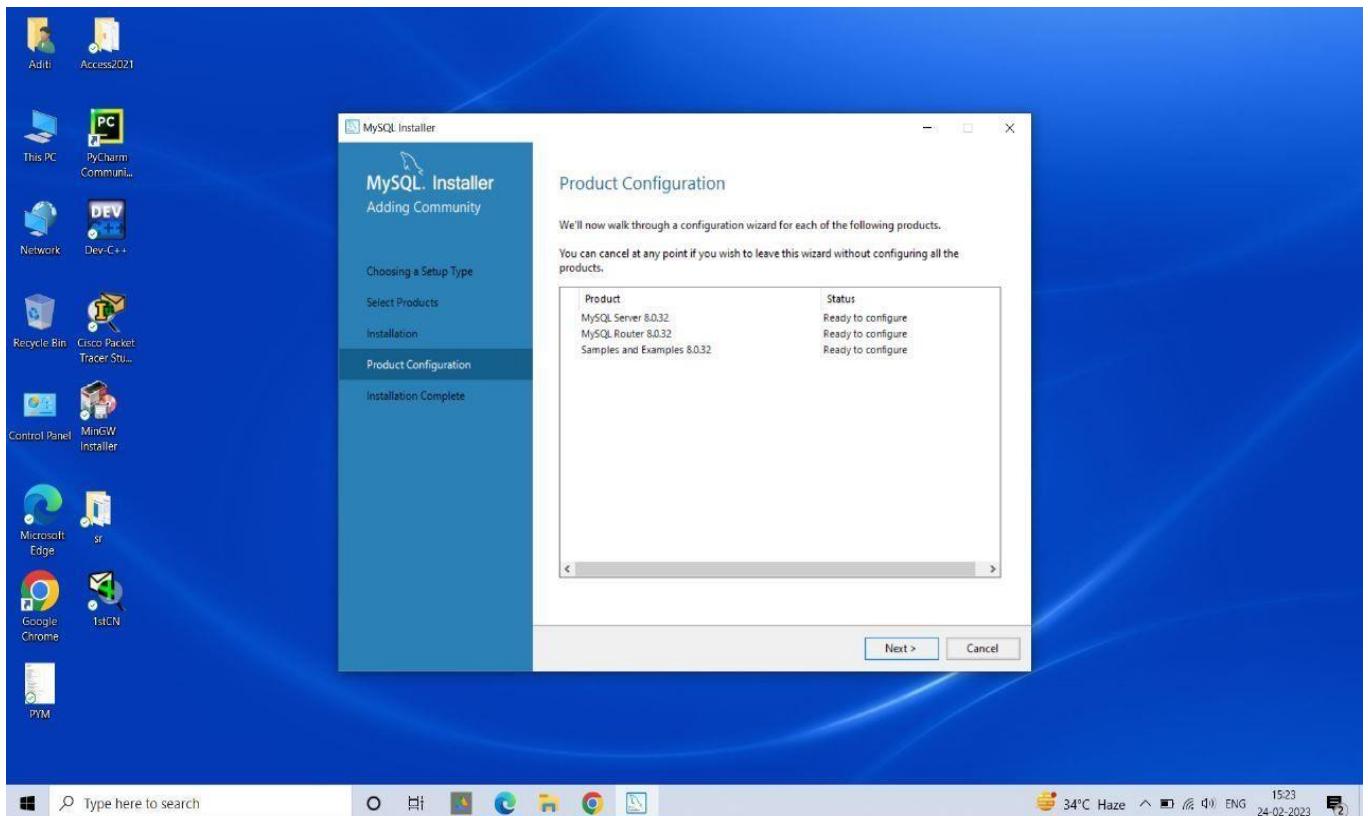
STEP 9:



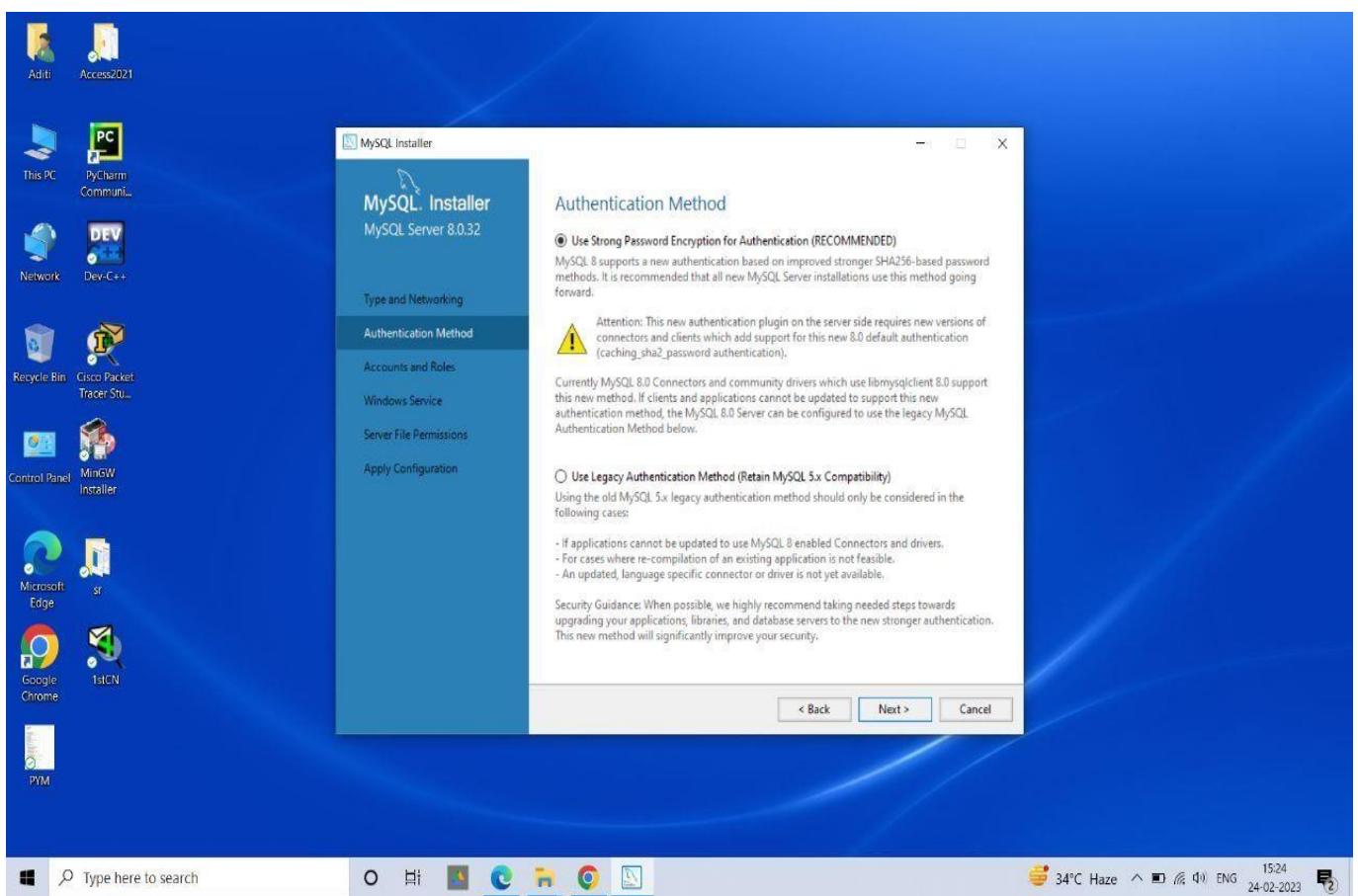
STEP 10:



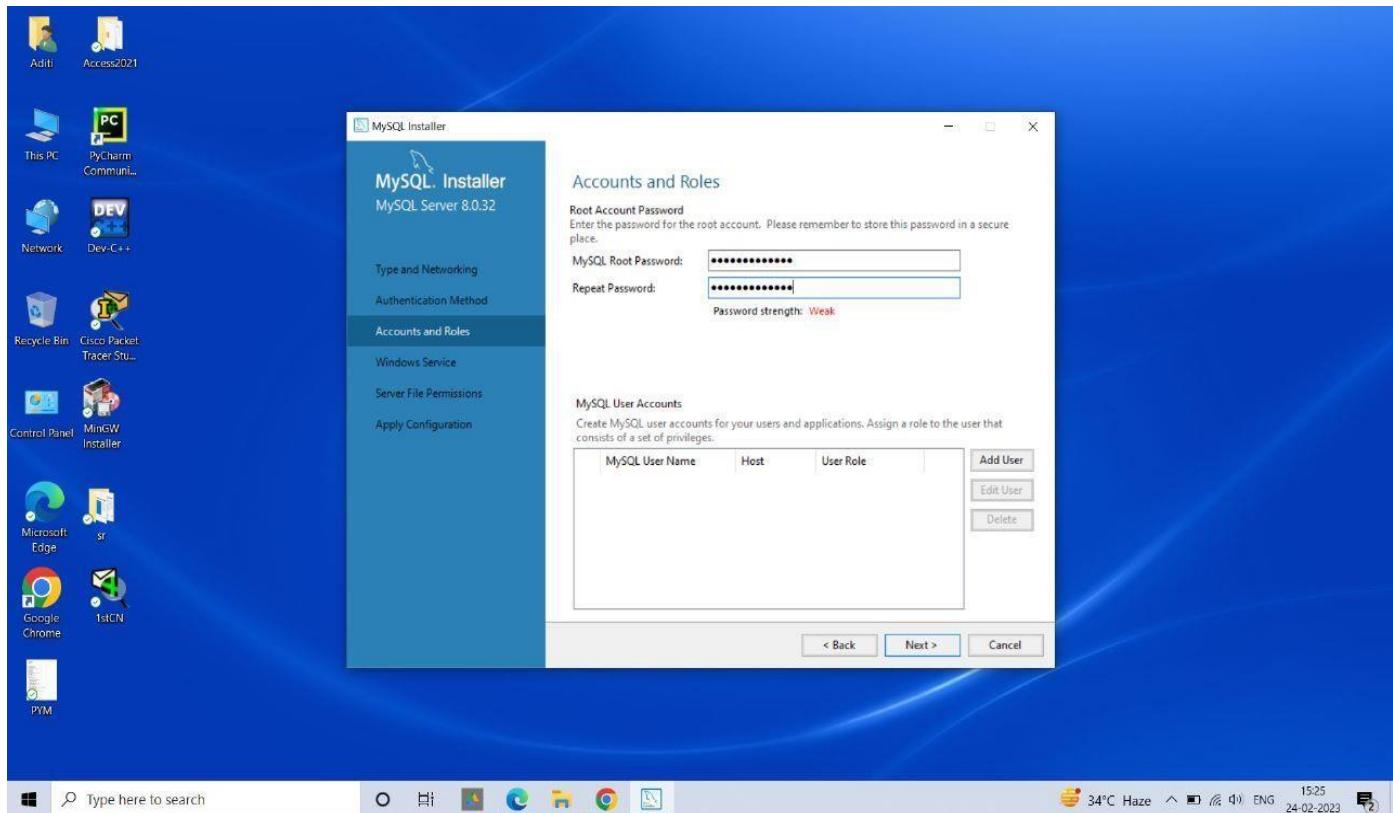
STEP 11:



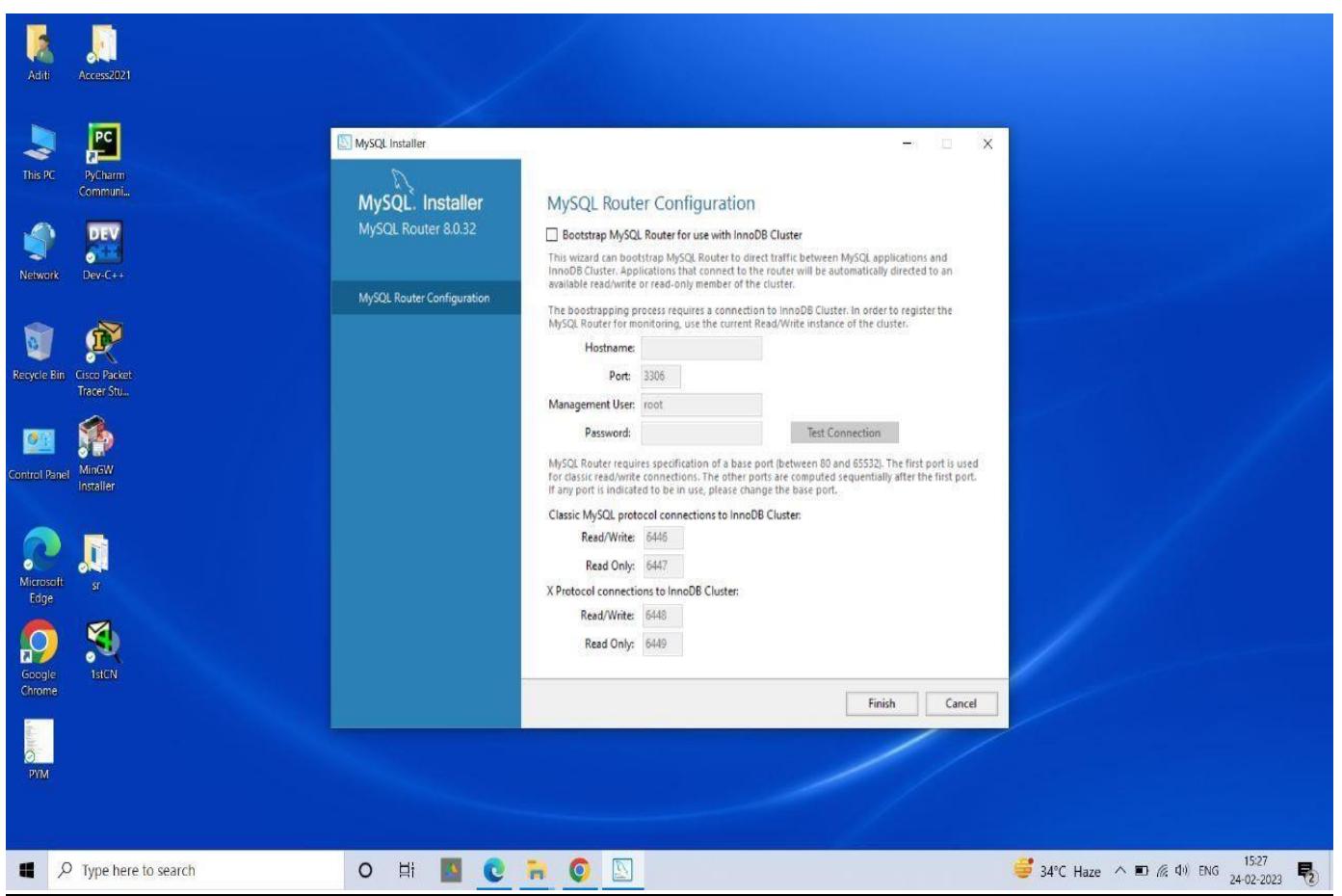
STEP 12:



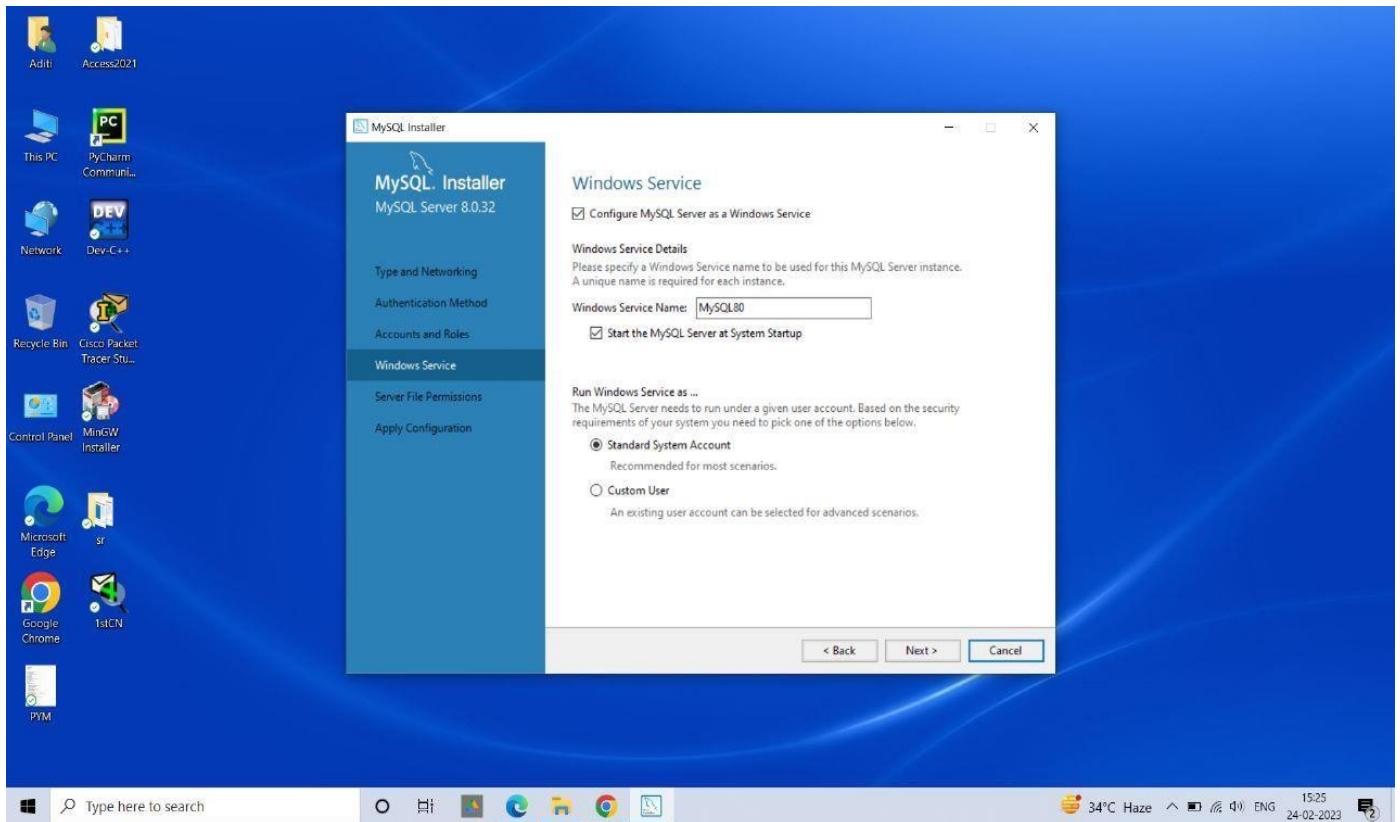
STEP 13:



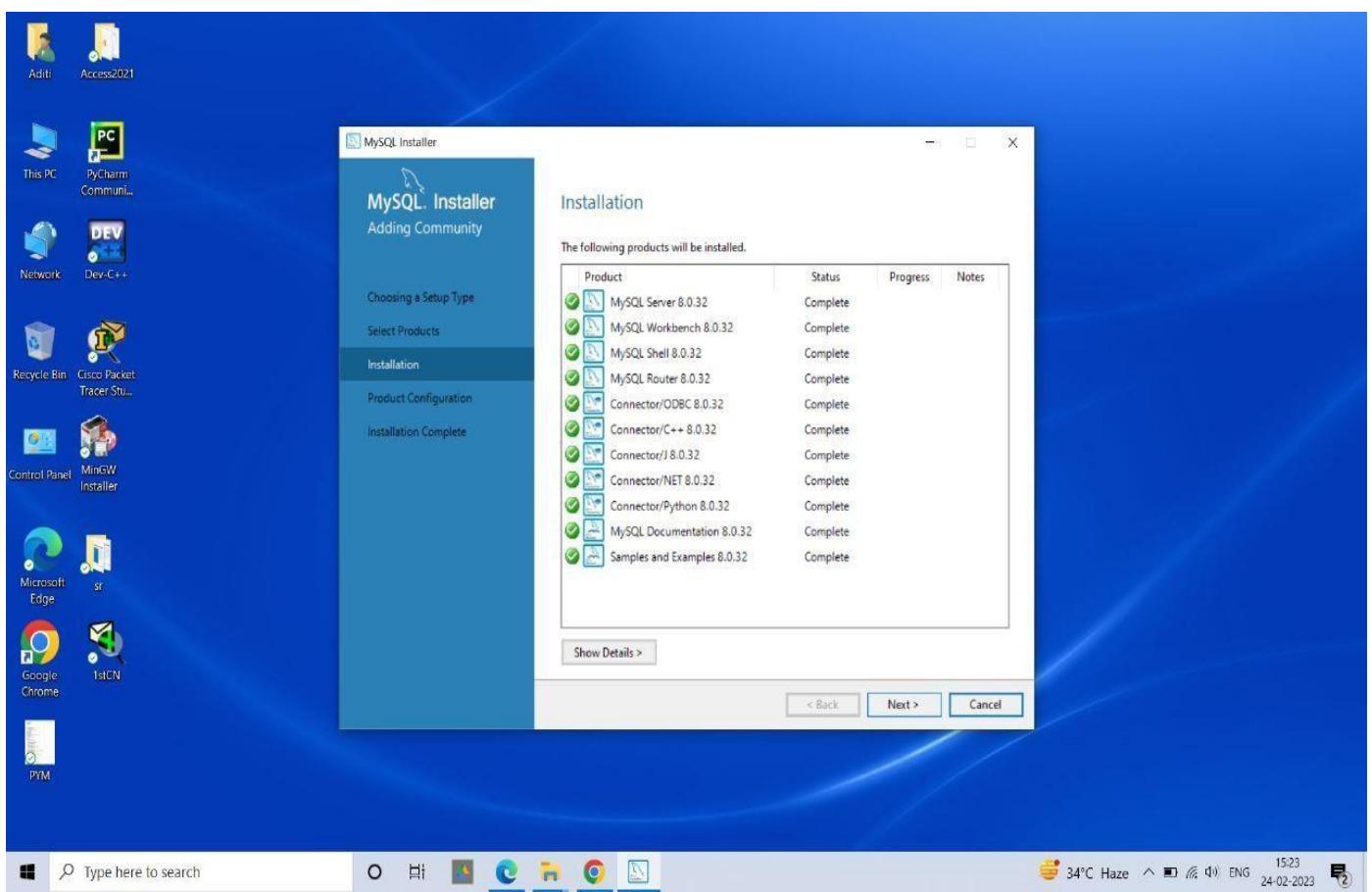
STEP 14:



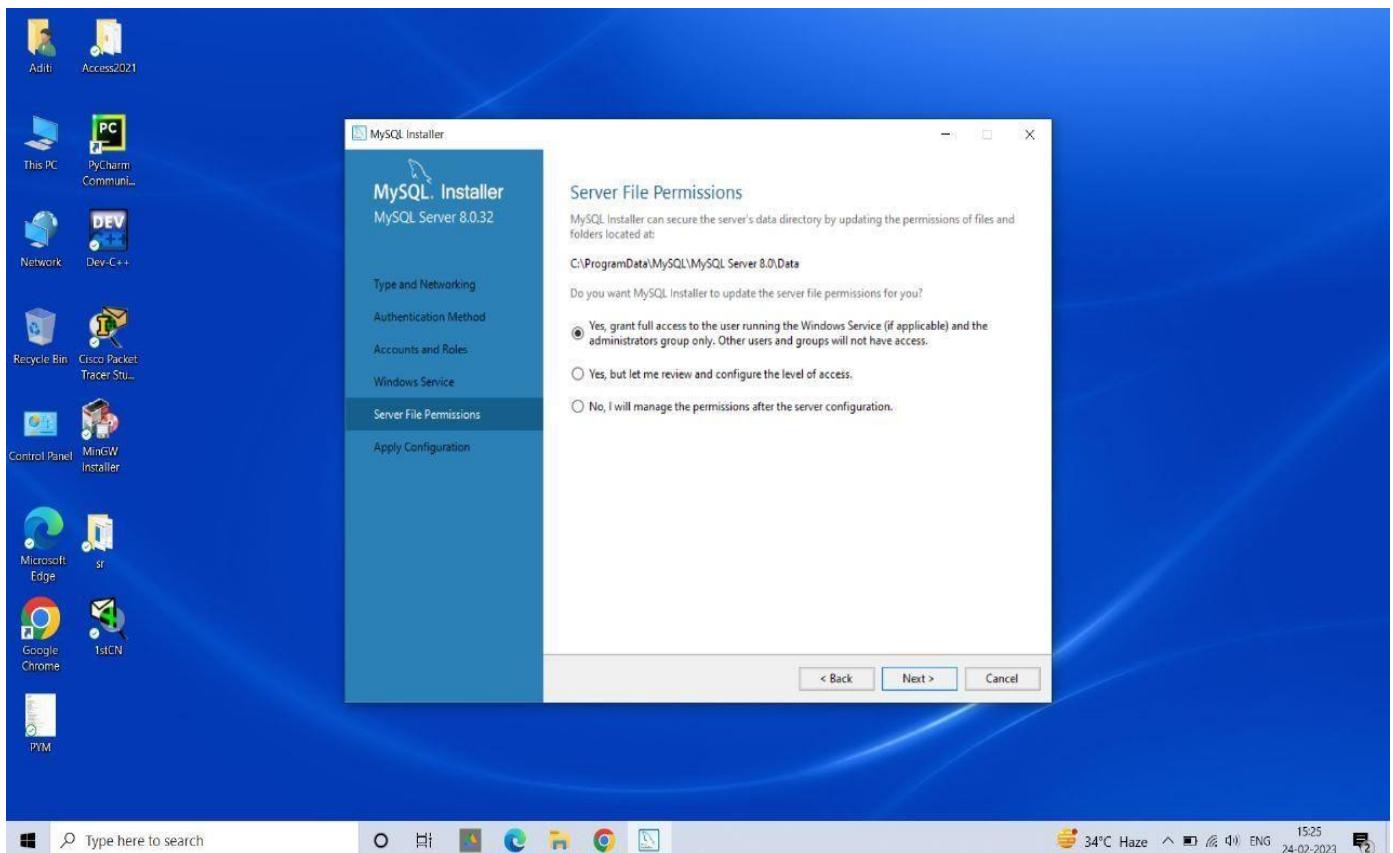
STEP 15:



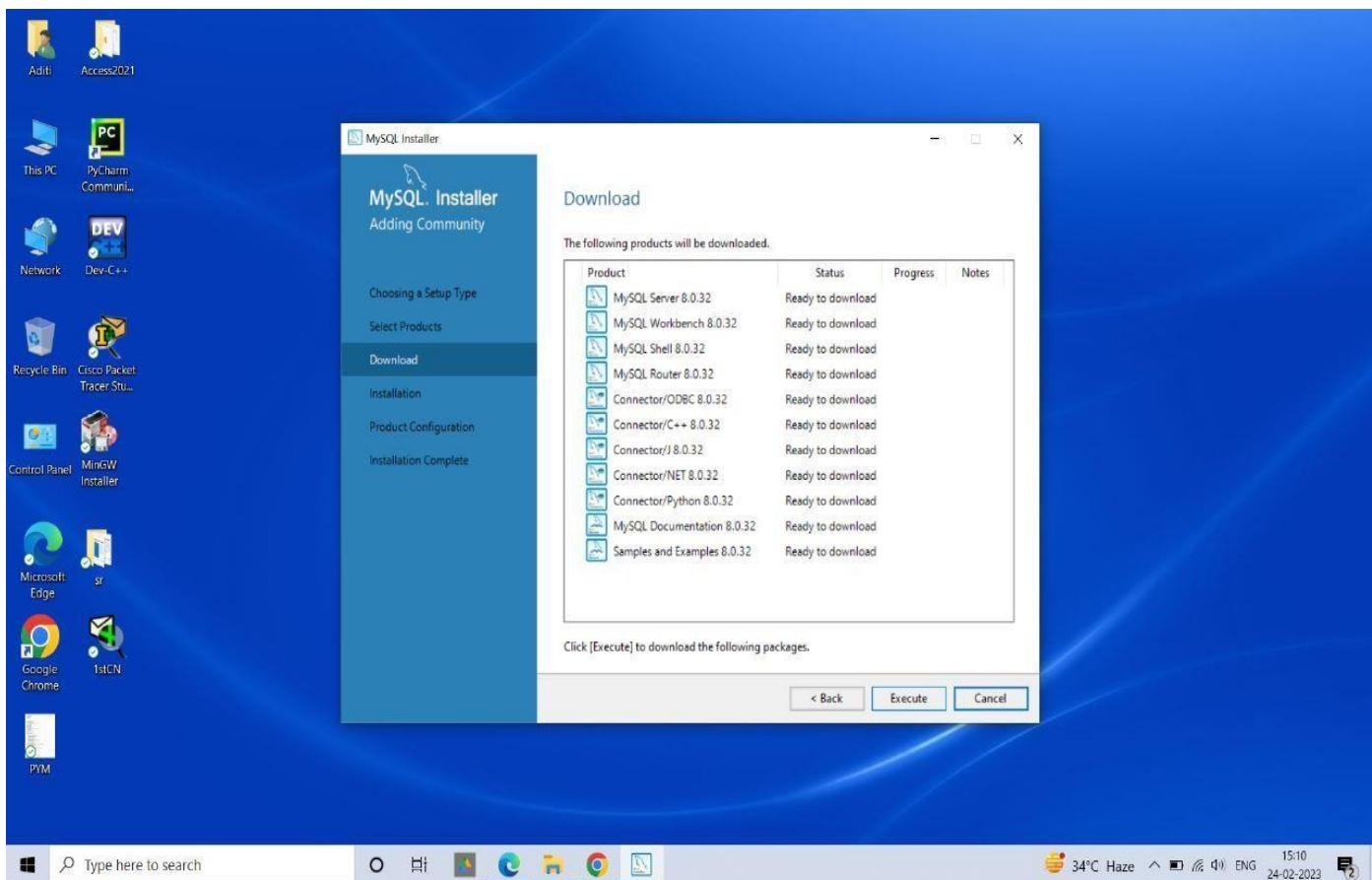
STEP 16:



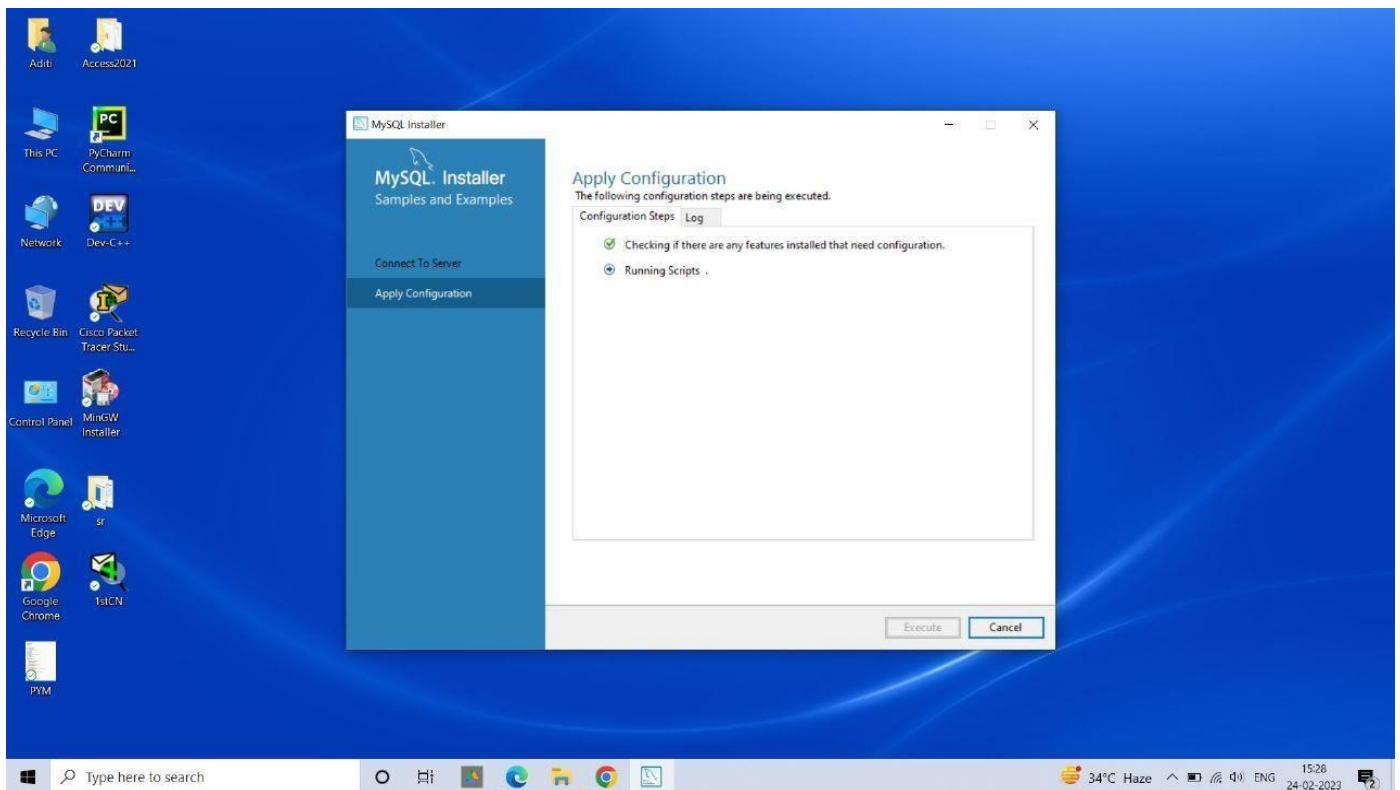
STEP 17:



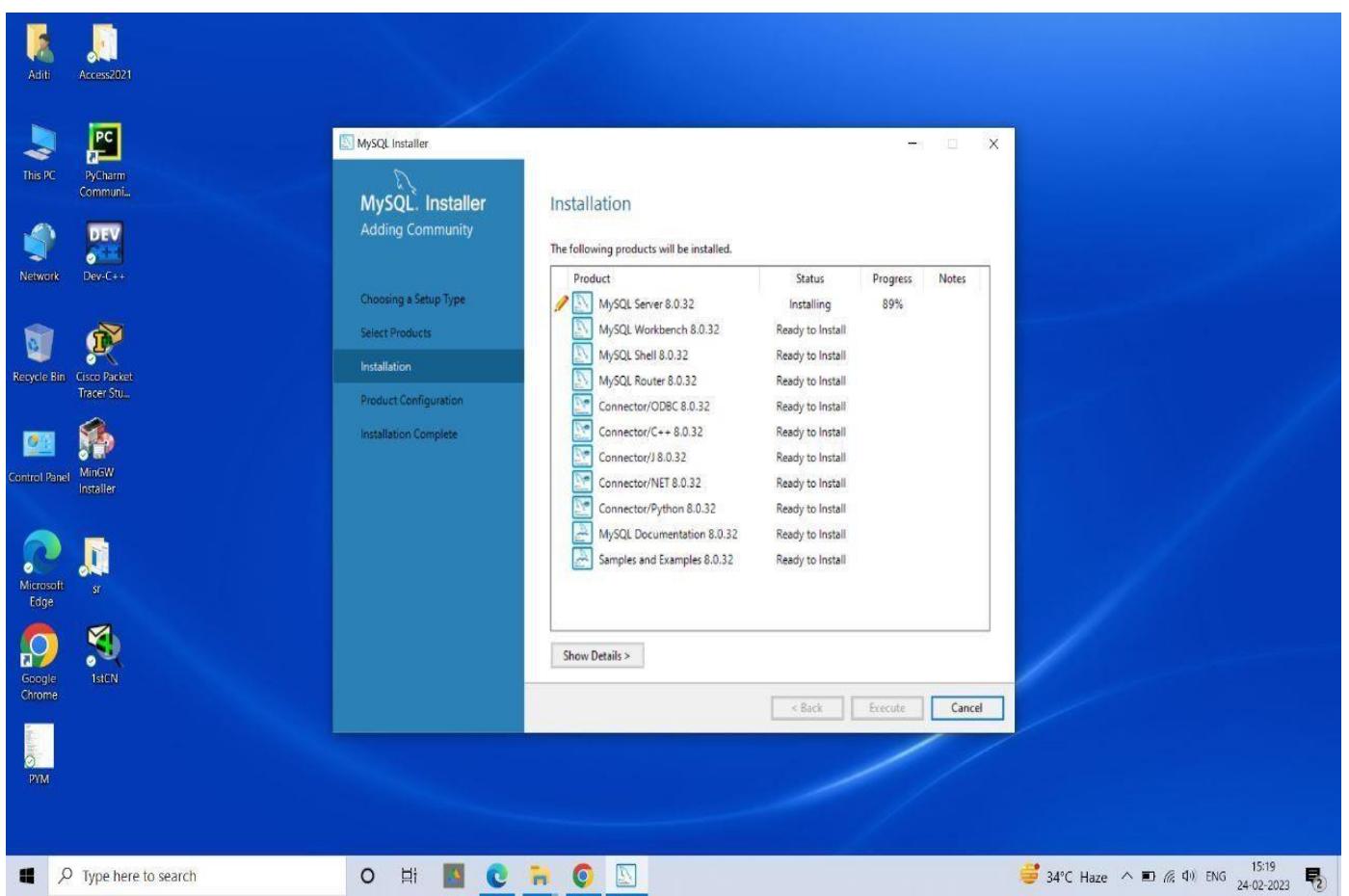
STEP 18:



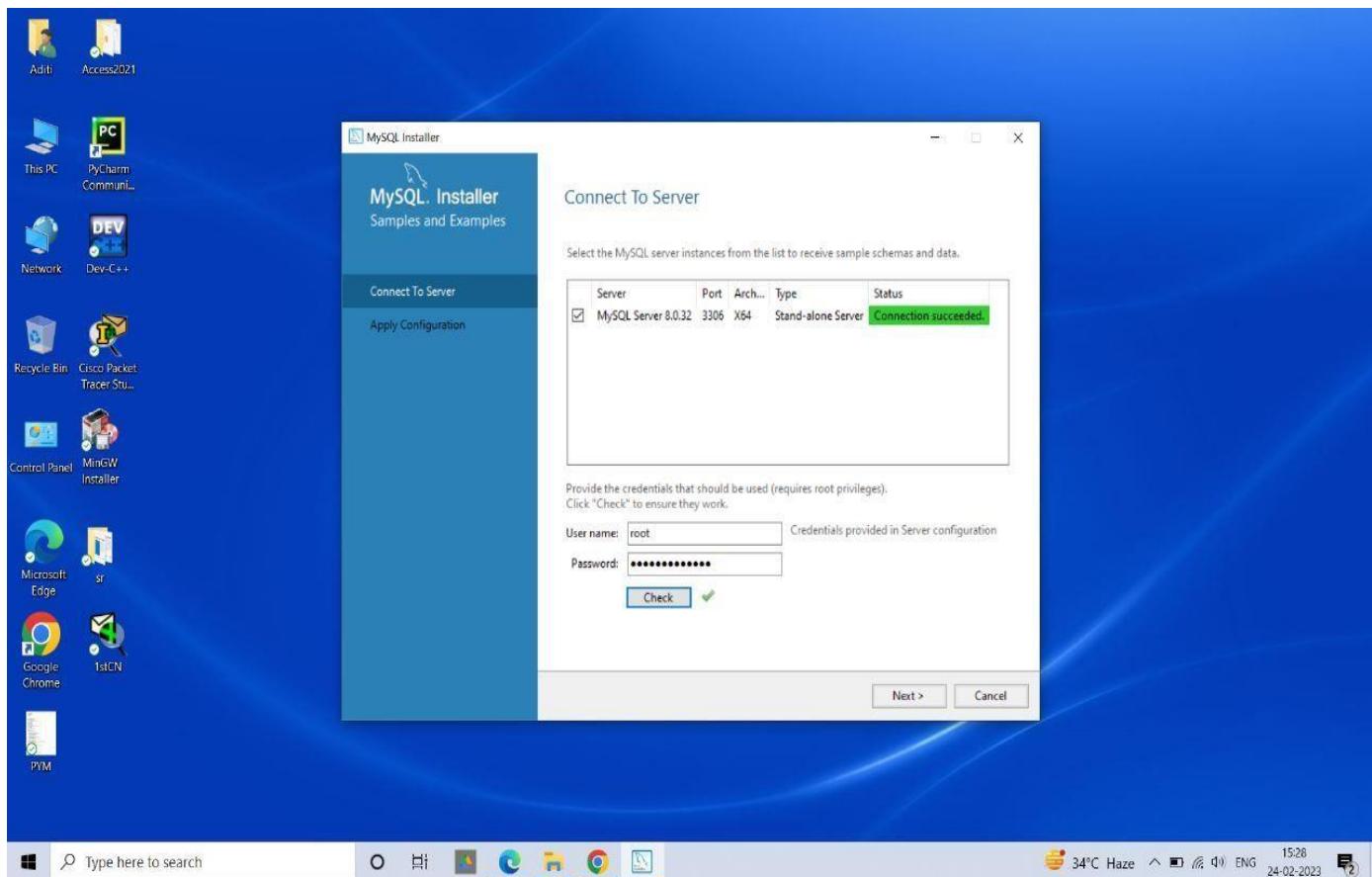
STEP 19:



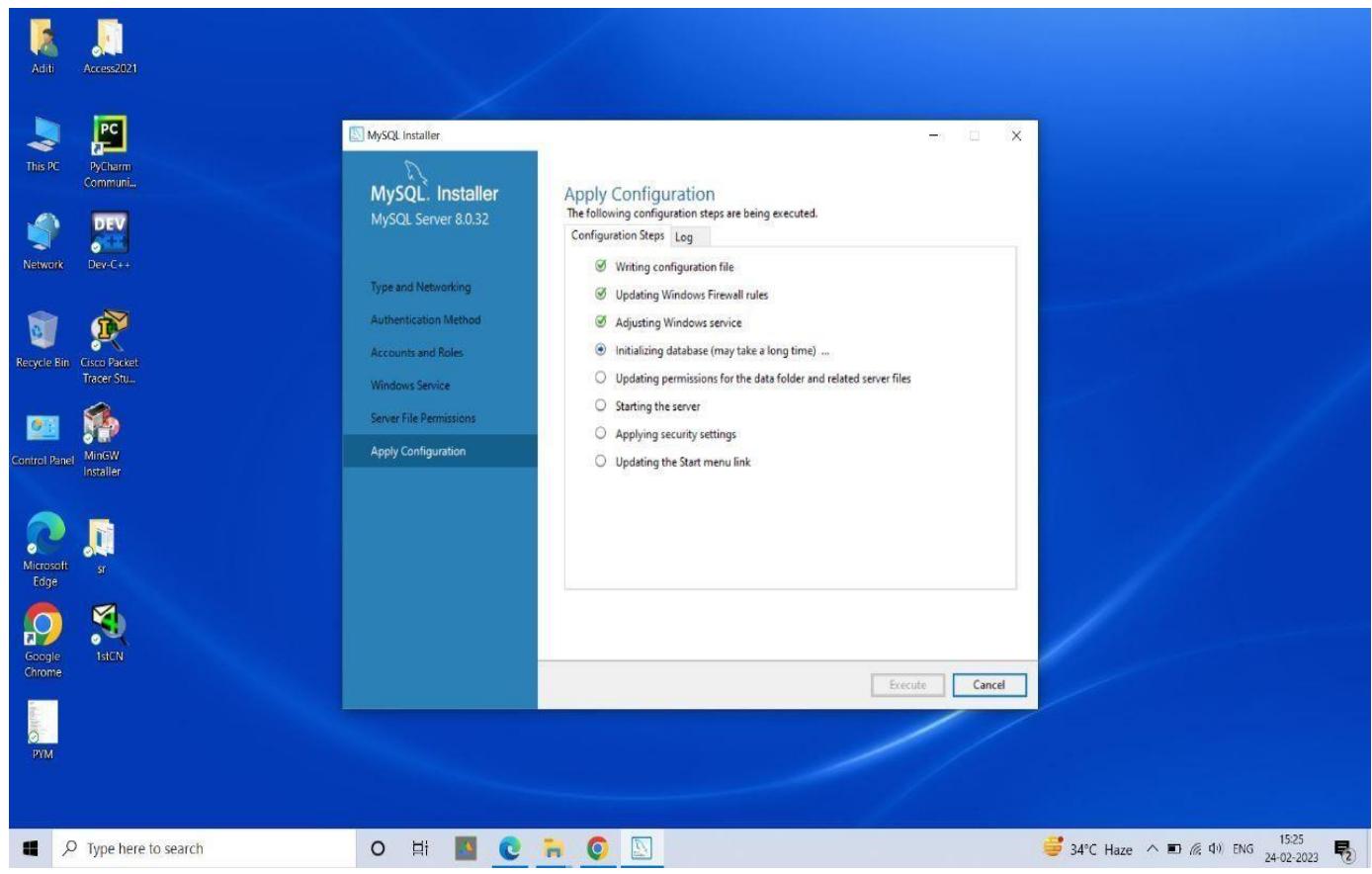
STEP 20:



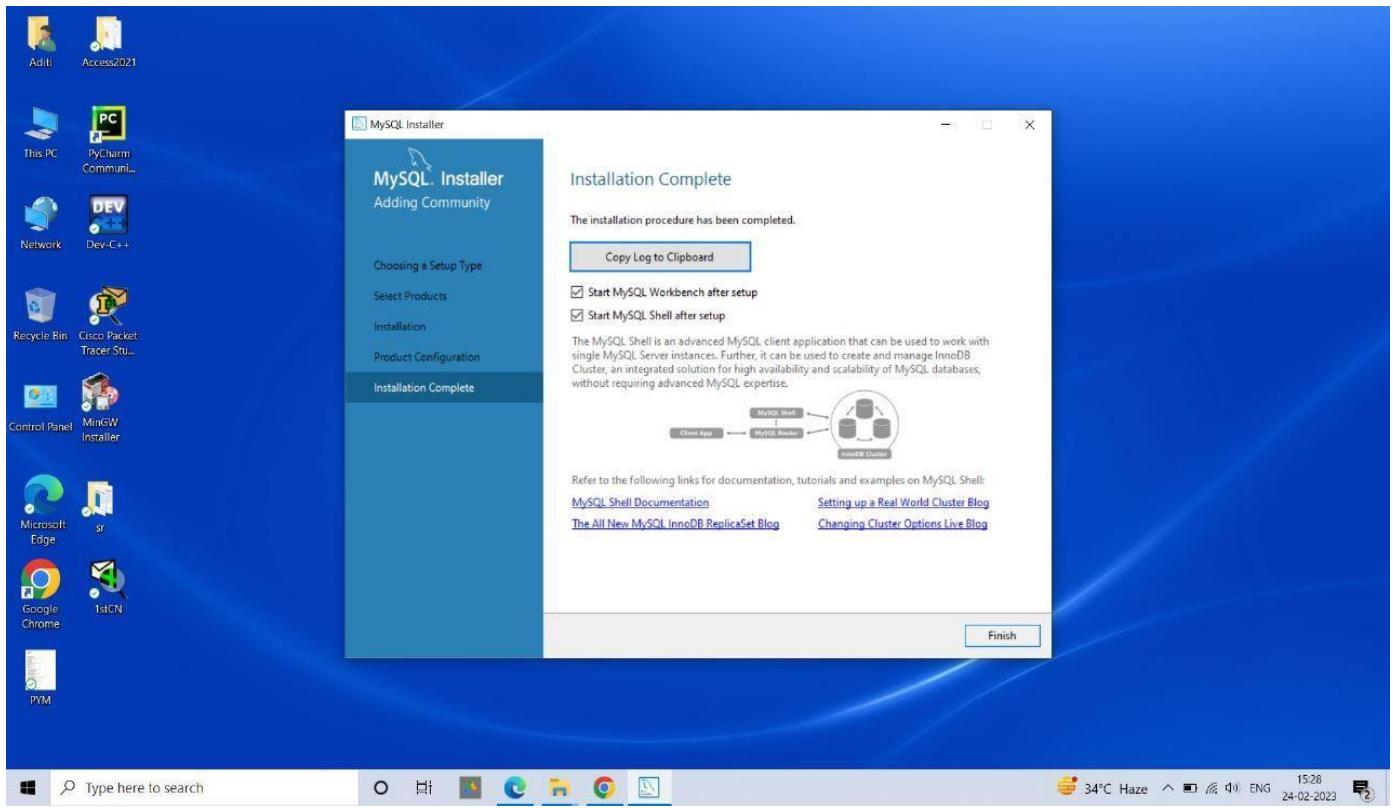
STEP 21:



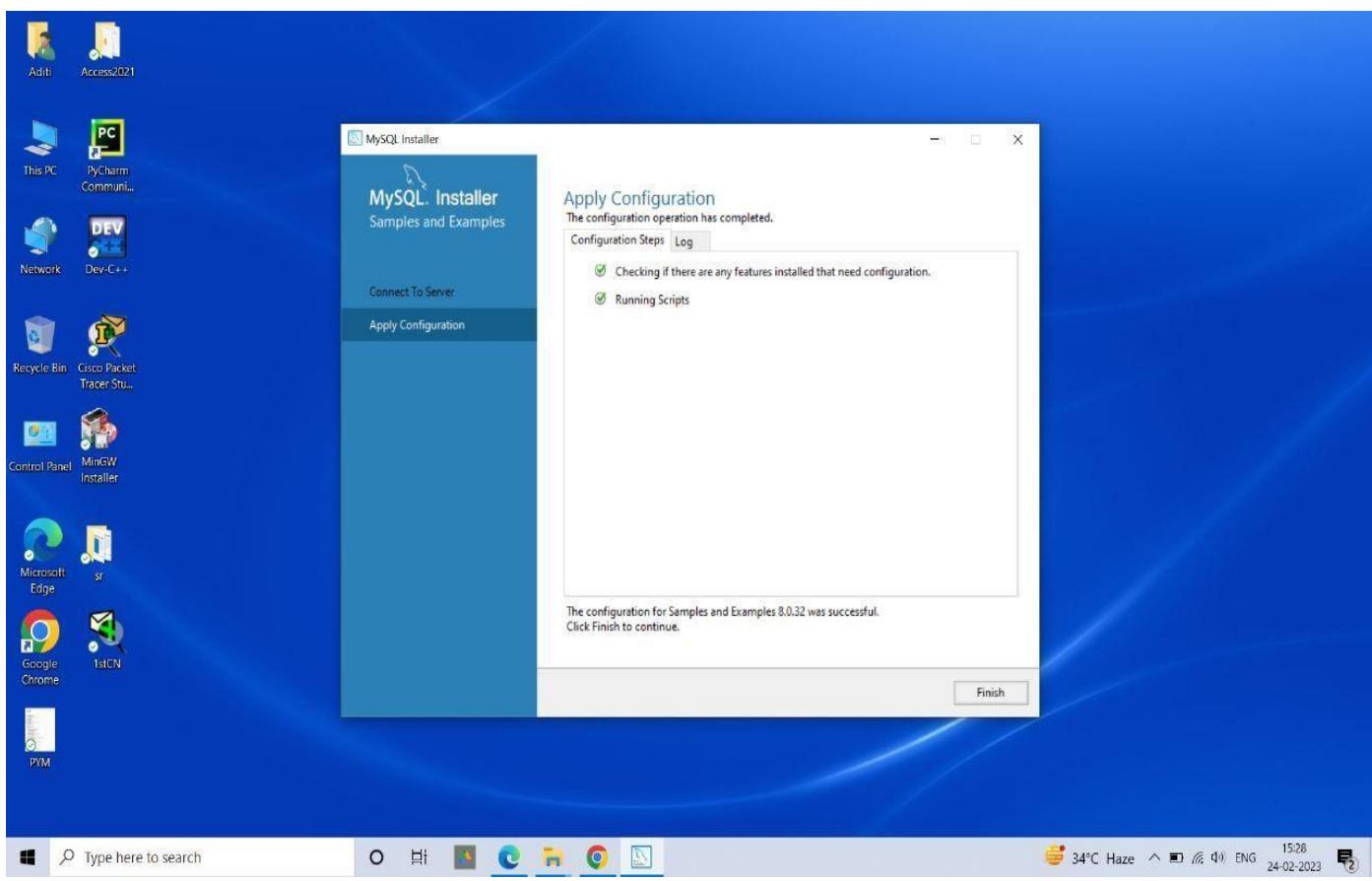
STEP 22:



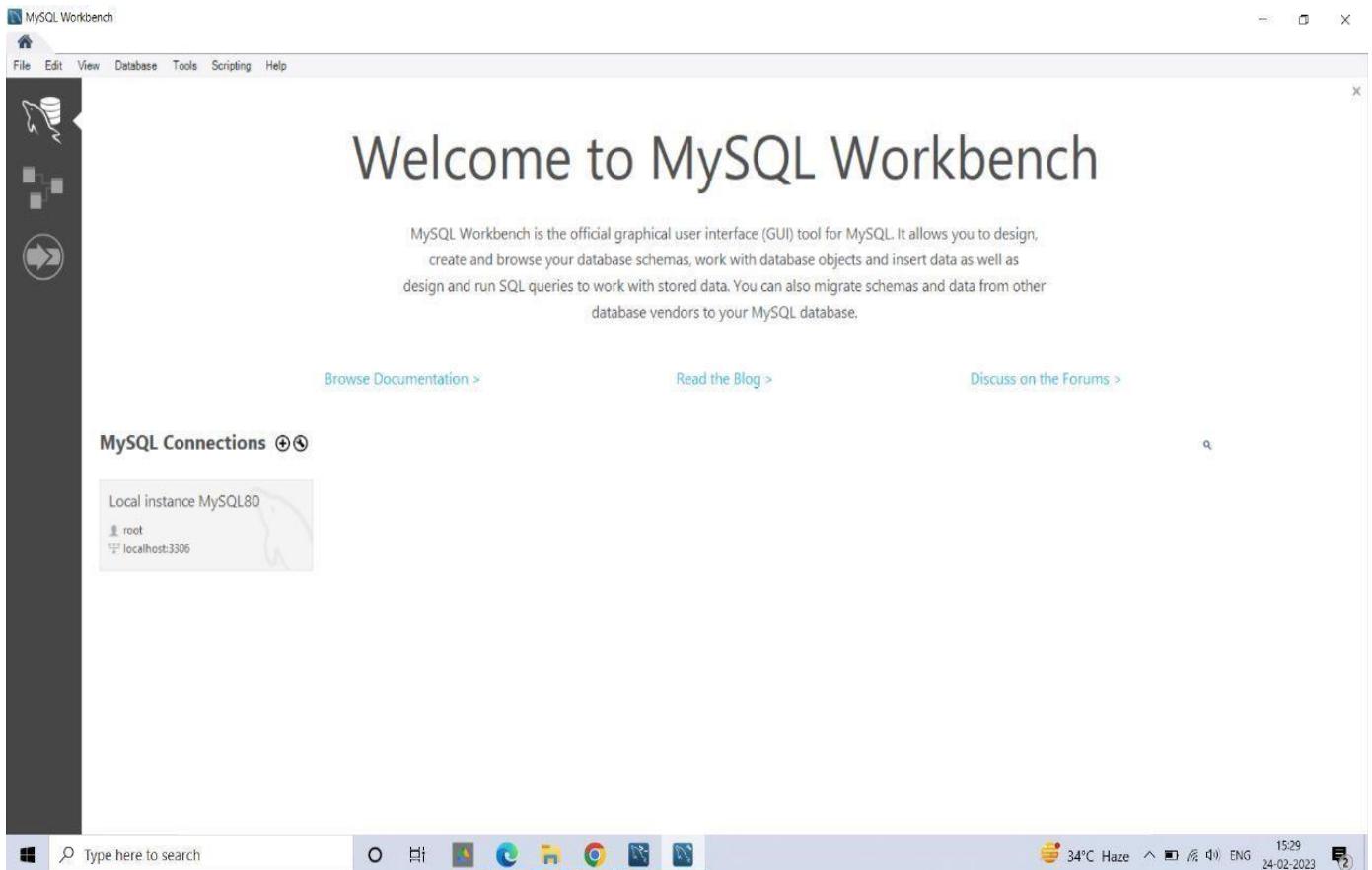
STEP 23:



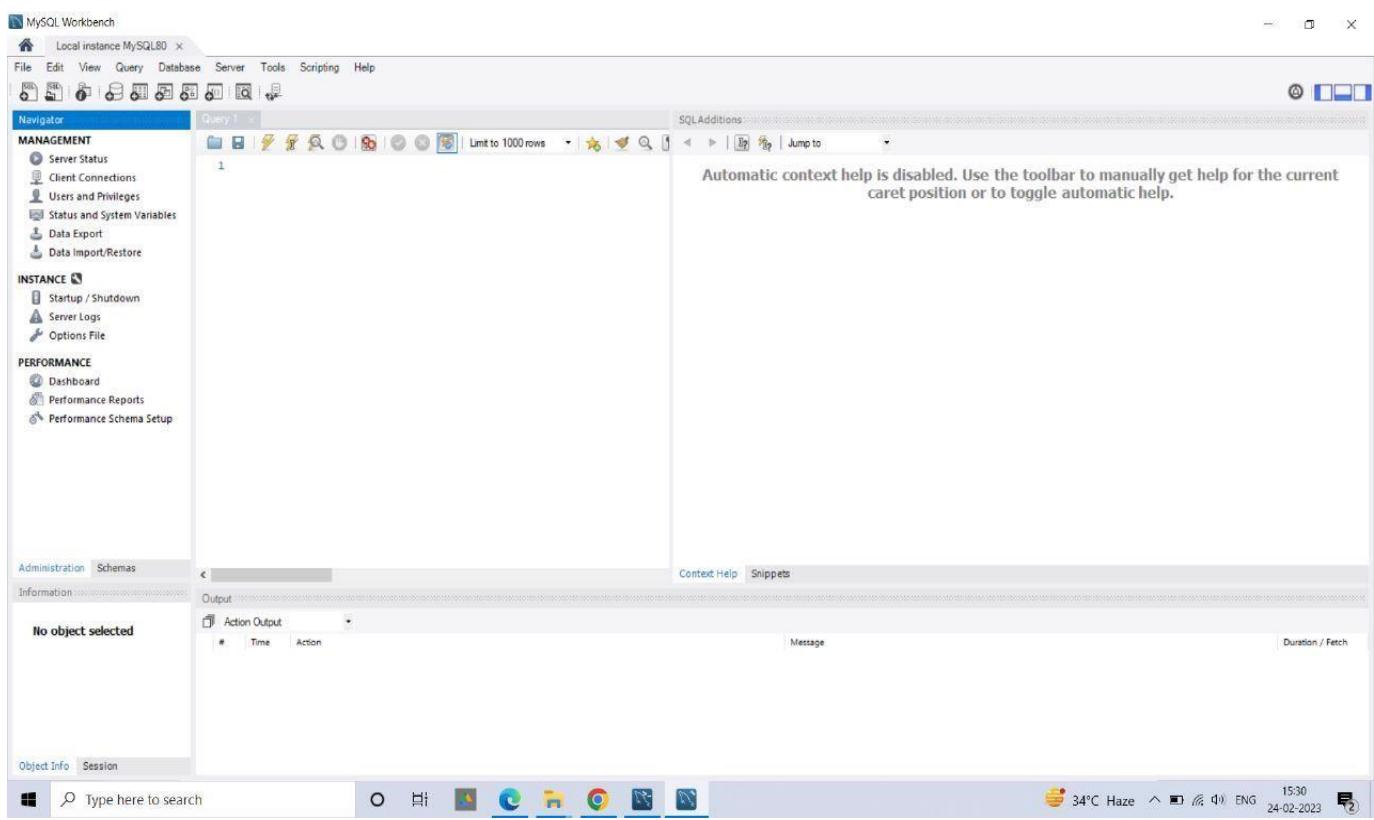
STEP 24:



STEP 25:



STEP 26:



STEP 27:

The screenshot shows the MySQL Workbench interface. In the top-left corner, the title bar reads "MySQL Workbench Local instance MySQL80". The main window has a toolbar at the top with various icons for file operations, navigation, and database management. Below the toolbar is a "Navigator" pane on the left containing sections for MANAGEMENT, INSTANCE, and PERFORMANCE. The central area is a "Query 1" editor with the following SQL code:

```
2 • use clg;
3 • create table clg_info
4 (clg_id int primary key,clg_name varchar (10),clg_address varchar(10));
5 • insert into clg_info values(1,'GPA','AWASARI');
6 • insert into clg_info values(2,'GPN','NASHIK');
7 • insert into clg_info values(3,'GPP','PUNE');
8 • select *from clg_info;
```

Below the SQL code is a "Result Grid" showing the data inserted into the table:

dg_id	clg_name	clg_address
1	GPA	AWASARI
2	GPN	NASHIK
3	GPP	PUNE

At the bottom of the interface, there is a "Session" tab and a taskbar with various application icons.

Group A: Study of Databases

3. Study of SQLite: What is SQLite? Uses of SQLite. Building and installing SQLite.

STEP 1:

The screenshot shows a web browser window with the Yahoo India homepage as the background. A search bar at the top contains the query "sqlite". Below the search bar, the results page displays approximately 29,500,000 search results. The first result is a link to the "SQLite Home Page" on sqlite.org. The page content includes sections like "Introduction", "Getting Started", "About", "Appropriate Uses for Sqlite", "JSON Functions", and "The TCL Interface Spec". To the right of the search results, there is a sidebar with the SQLite logo and a brief description: "SQLite is a serverless relational database management system (RDBMS)". Below this, there is a link to "sqlite.org" and a "Wiki" section. The browser's address bar shows the URL <https://in.search.yahoo.com/search?fr=mcafee&type=E210IN826G91769&p=sqlite>. The taskbar at the bottom of the screen shows various application icons.

STEP 2:

The screenshot shows the official SQLite website homepage (<https://www.sqlite.org/index.html>). The page features the SQLite logo and navigation links for Home, About, Documentation, Download, License, Support, and Purchase. The main content area discusses what SQLite is, its stability, and its widespread use. It also mentions the file format, source code availability, and the latest release (Version 3.41.2). On the right side, there is a "Common Links" sidebar containing a list of links related to SQLite's features, syntax, interface specifications, and frequently asked questions. The browser's address bar shows the URL <https://www.sqlite.org/index.html>. The taskbar at the bottom of the screen shows various application icons.

STEP 3:

Precompiled Binaries for Linux

- [sqlite-tools-linux-x86-3410200.zip](#) (3.26 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher](#) program, and the [sqlite3_analyzer](#) program.
(SHA3-256: 6cd0ca943868978a945d4531bcd4e21fee8e3a7650e577f052550a37bc62db6e)
(2.16 MiB)

Precompiled Binaries for Mac OS X (x86)

- [sqlite-tools-osx-x86-3410200.zip](#) (1.54 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher](#) program, and the [sqlite3_analyzer](#) program.
(SHA3-256: e0e01eeade6ff7bc6f0ddbf0fe86798a595494328e1afa15da192e5fc516cb29)

Precompiled Binaries for Windows

- [sqlite-dll-win32-x86-3410200.zip](#) (563.08 KIB) 32-bit DLL (x86) for SQLite version 3.41.2.
(SHA3-256: 2a3b03595ee7c3b172465d9be3990fd8ac10fa93b9af22031b67fa42c2325e1)
- [sqlite-dll-win64-x64-3410200.zip](#) (1.15 MiB) 64-bit DLL (x64) for SQLite version 3.41.2.
(SHA3-256: a85aa4c03f394147b4a9050245fb8d5d0b00ae726d1f259e71c07055e44854cf)
- [sqlite-tools-win32-x86-3410200.zip](#) (1.91 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher.exe](#) program, and the [sqlite3_analyzer.exe](#) program.
(SHA3-256: 0ceebb7f8378707d6d6b0737ecdf2ba02253a3b44b1009400f86273719d98f1f)

Precompiled Binaries for .NET

- [System.Data.SQLite](#) Visit the [System.Data.SQLite.org](#) website and especially the [download page](#) for source code and binaries of SQLite for .NET.

WebAssembly & JavaScript

- [sqlite-wasm-3410200.zip](#) (702.24 KIB) A precompiled bundle of [sqlite3.wasm](#) and its JavaScript APIs, ready for use in web applications.
(SHA3-256: 1713c3a4c25b09f5d52ed25fb81948f6081e9eefb5a6d449ae1cd848e962c)

Alternative Source Code Formats

- [sqlite-src-3410200.zip](#) (13.20 MiB) Snapshot of the complete (raw) source tree for SQLite version 3.41.2. See [How To Compile SQLite](#) for usage details.
(SHA3-256: 793e24c5158bafdb8a6e861ea9ad267cc773705d0beb78b5c4aa323033e8028)
- [sqlite-preprocessed](#) Preprocessed C sources for SQLite version 3.41.2.

STEP 4:

Precompiled Binaries for Linux

- [sqlite-tools-linux-x86-3410200.zip](#) (3.26 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher](#) program, and the [sqlite3_analyzer](#) program.
(SHA3-256: 6cd0ca943868978a945d4531bcd4e21fee8e3a7650e577f052550a37bc62db6e)
(2.16 MiB)

Precompiled Binaries for Mac OS X (x86)

- [sqlite-tools-osx-x86-3410200.zip](#) (1.54 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher](#) program, and the [sqlite3_analyzer](#) program.
(SHA3-256: e0e01eeade6ff7bc6f0ddbf0fe86798a595494328e1afa15da192e5fc516cb29)

Precompiled Binaries for Windows

- [sqlite-dll-win32-x86-3410200.zip](#) (563.08 KIB) 32-bit DLL (x86) for SQLite version 3.41.2.
(SHA3-256: 2a3b03595ee7c3b172465d9be3990fd8ac10fa93b9af22031b67fa42c2325e1)
- [sqlite-dll-win64-x64-3410200.zip](#) (1.15 MiB) 64-bit DLL (x64) for SQLite version 3.41.2.
(SHA3-256: a85aa4c03f394147b4a9050245fb8d5d0b00ae726d1f259e71c07055e44854cf)
- [sqlite-tools-win32-x86-3410200.zip](#) (1.91 MiB) A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqlcipher.exe](#) program, and the [sqlite3_analyzer.exe](#) program.
(SHA3-256: 0ceebb7f8378707d6d6b0737ecdf2ba02253a3b44b1009400f86273719d98f1f)

Precompiled Binaries for .NET

- [System.Data.SQLite](#) Visit the [System.Data.SQLite.org](#) website and especially the [download page](#) for source code and binaries of SQLite for .NET.

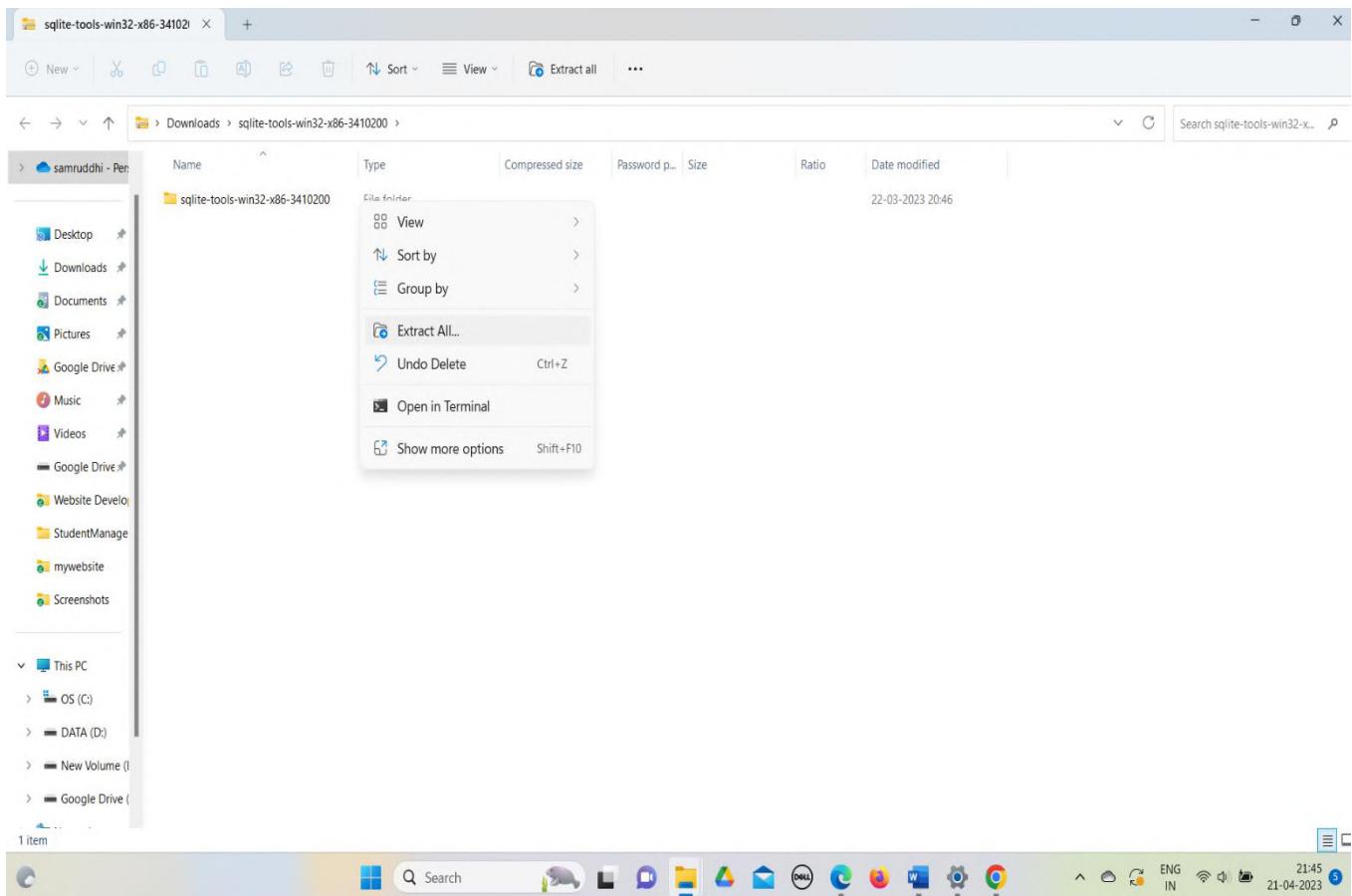
WebAssembly & JavaScript

- [sqlite-wasm-3410200.zip](#) (702.24 KIB) A precompiled bundle of [sqlite3.wasm](#) and its JavaScript APIs, ready for use in web applications.
(SHA3-256: 1713c3a4c25b09f5d52ed25fb81948f6081e9eefb5a6d449ae1cd848e962c)

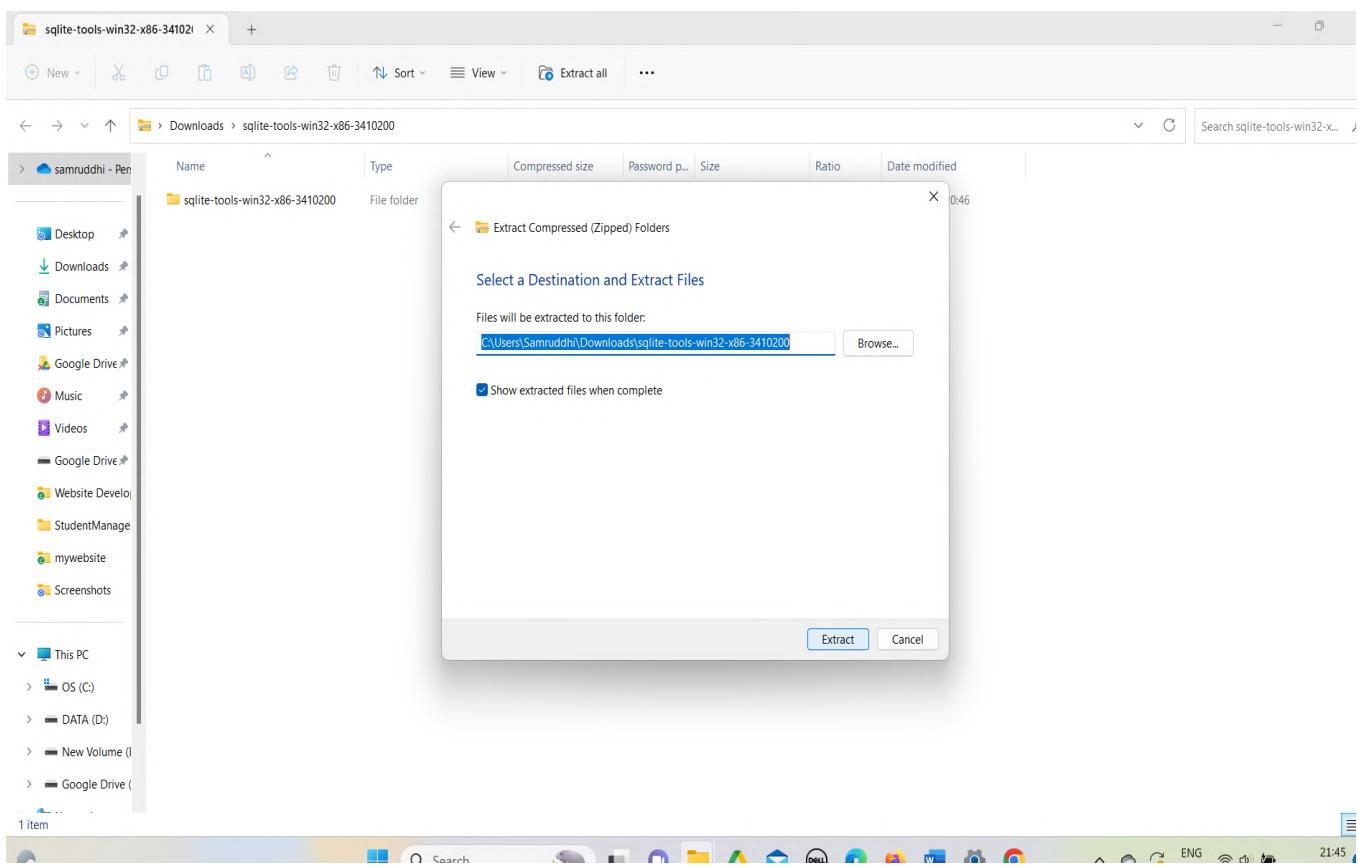
Alternative Source Code Formats

- [sqlite-src-3410200.zip](#) (13.20 MiB) Snapshot of the complete (raw) source tree for SQLite version 3.41.2. See [How To Compile SQLite](#) for usage details.
(SHA3-256: 793e24c5158bafdb8a6e861ea9ad267cc773705d0beb78b5c4aa323033e8028)

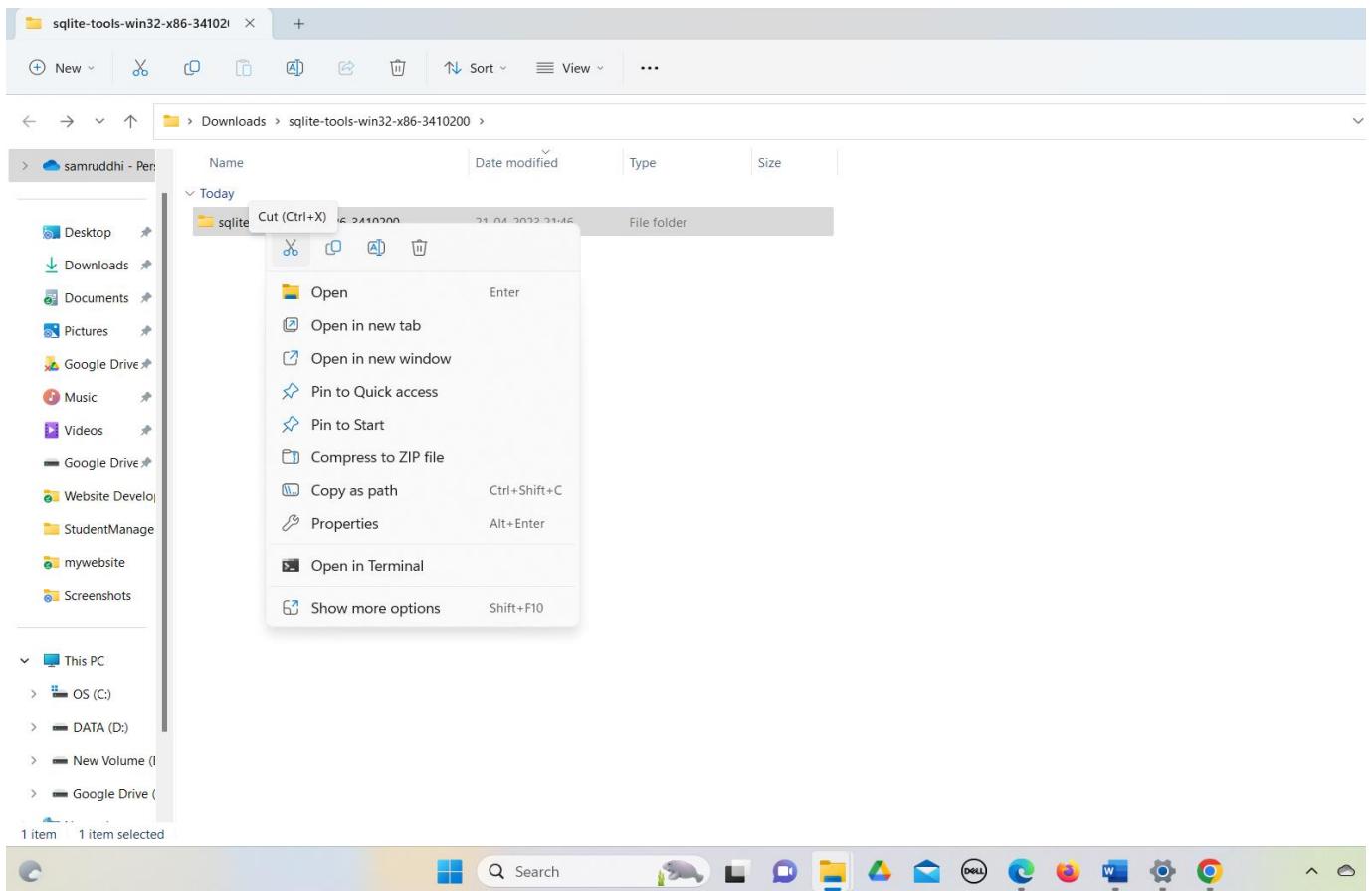
STEP 5:



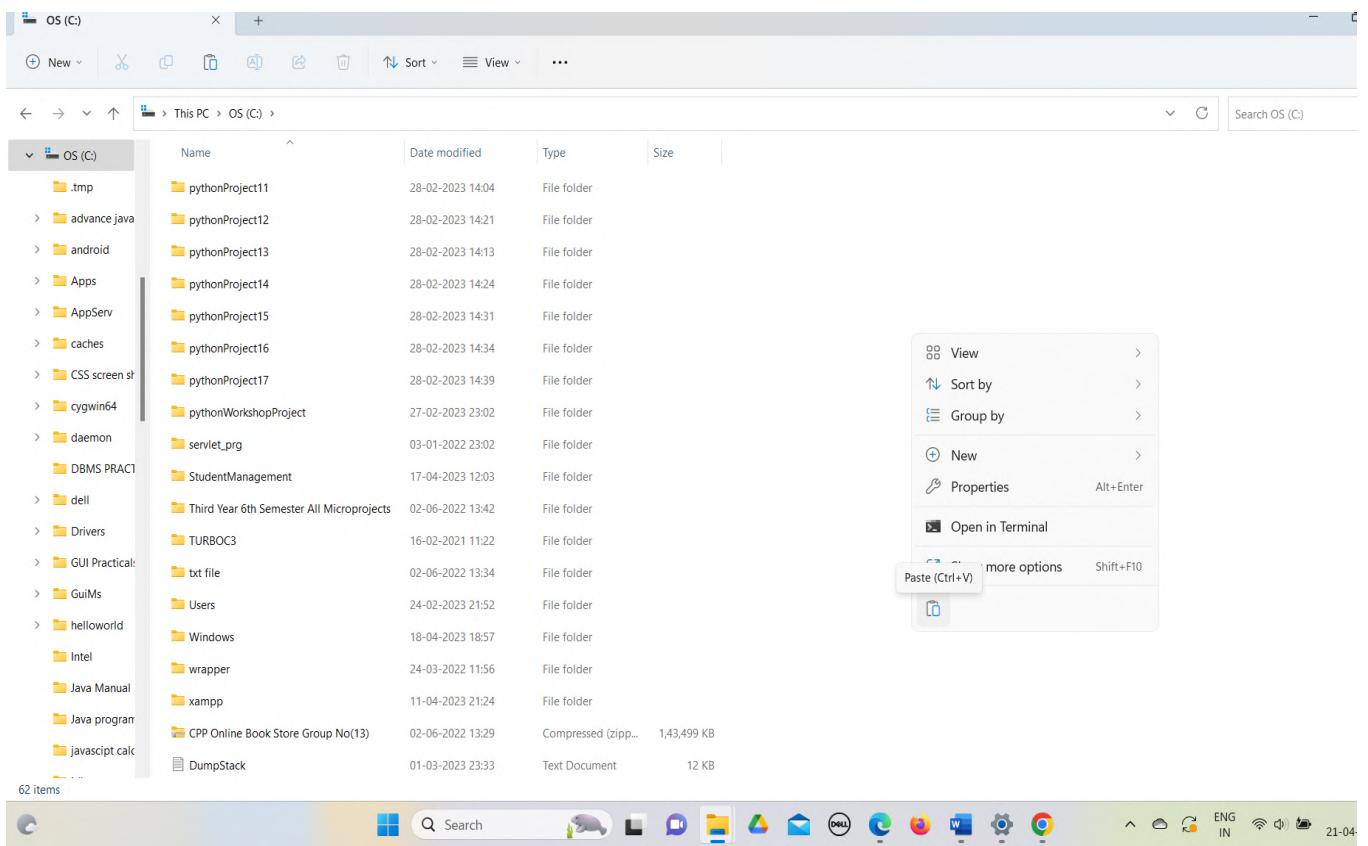
STEP 6:



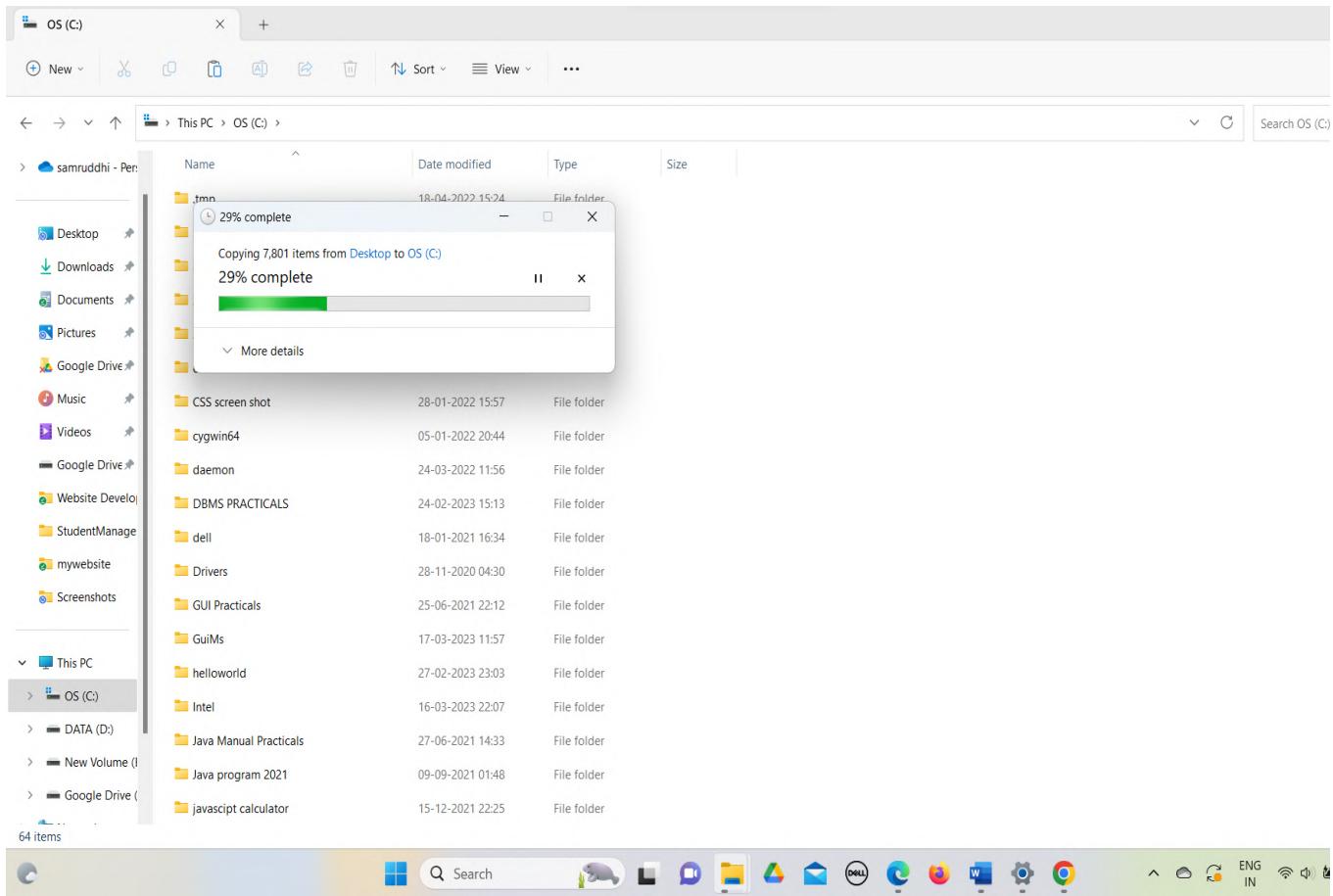
STEP 7:



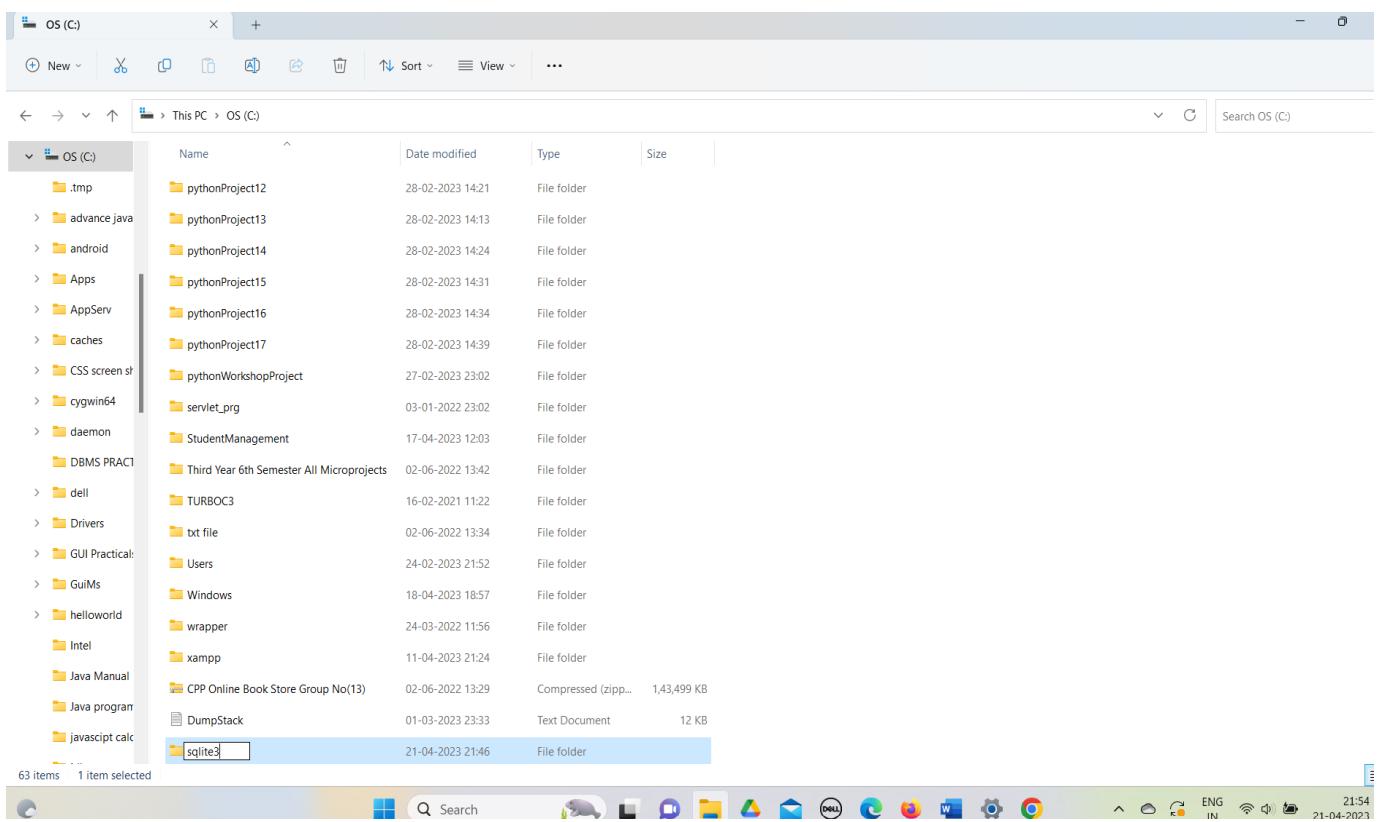
STEP 8:



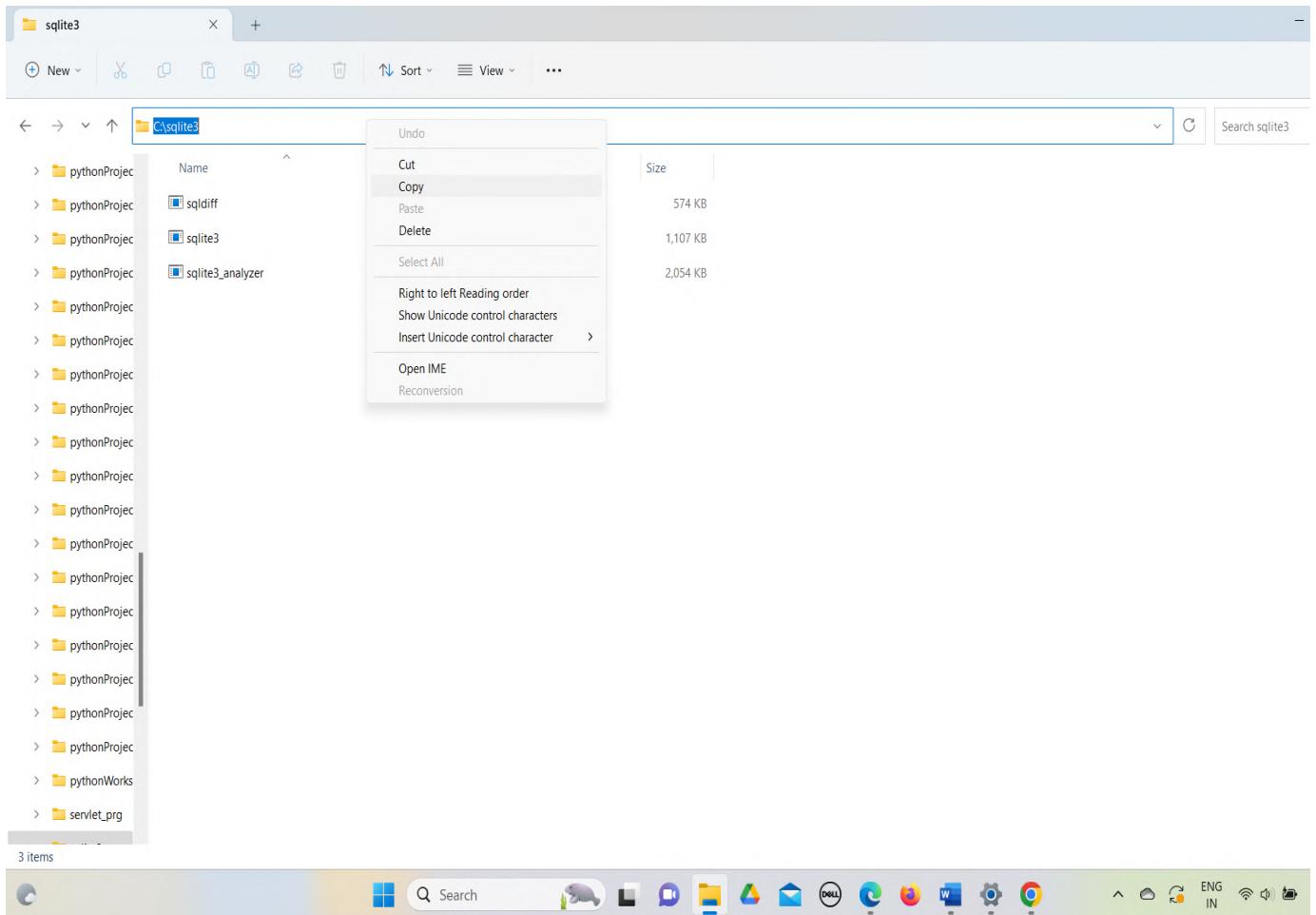
STEP 9:



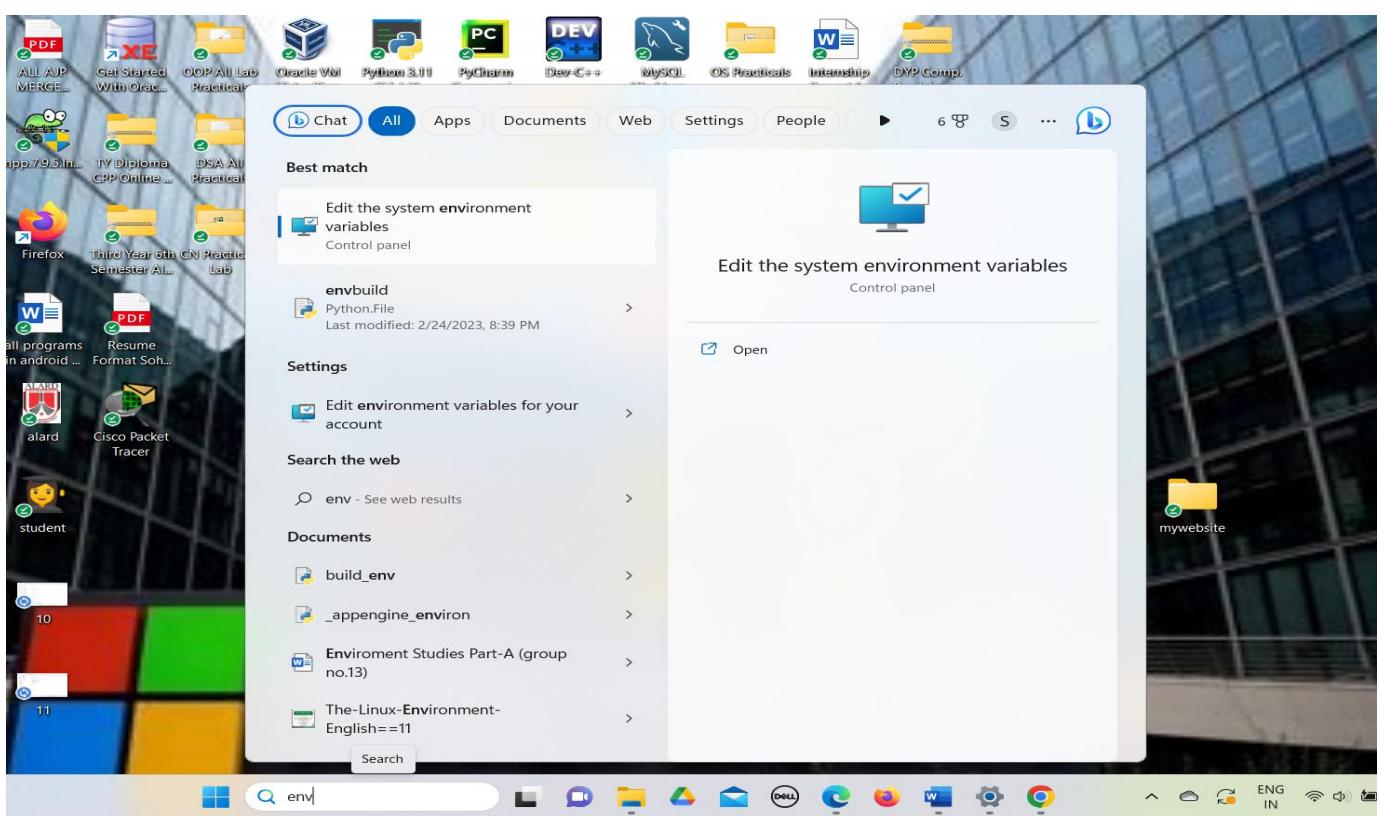
STEP 10:



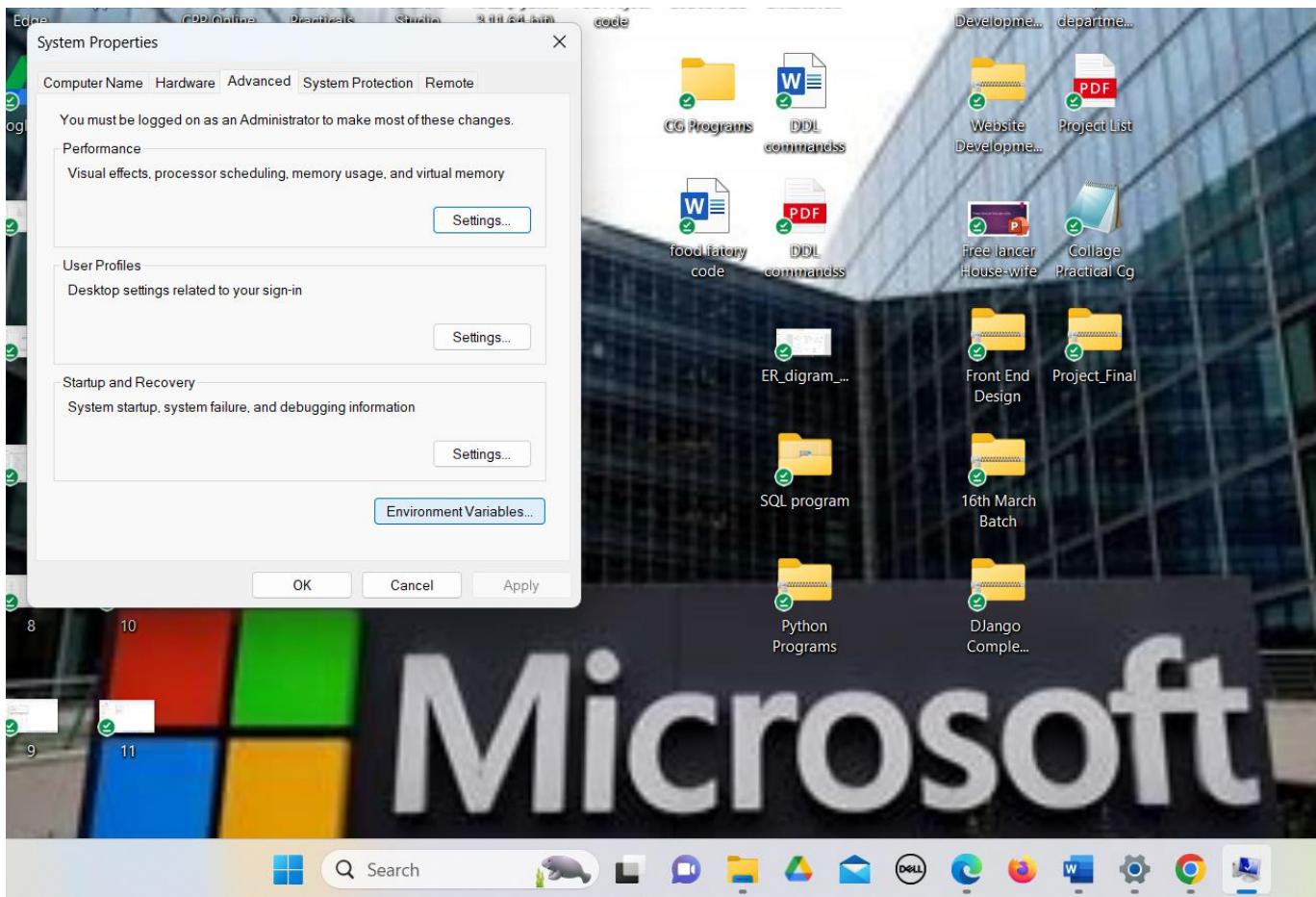
STEP 11:



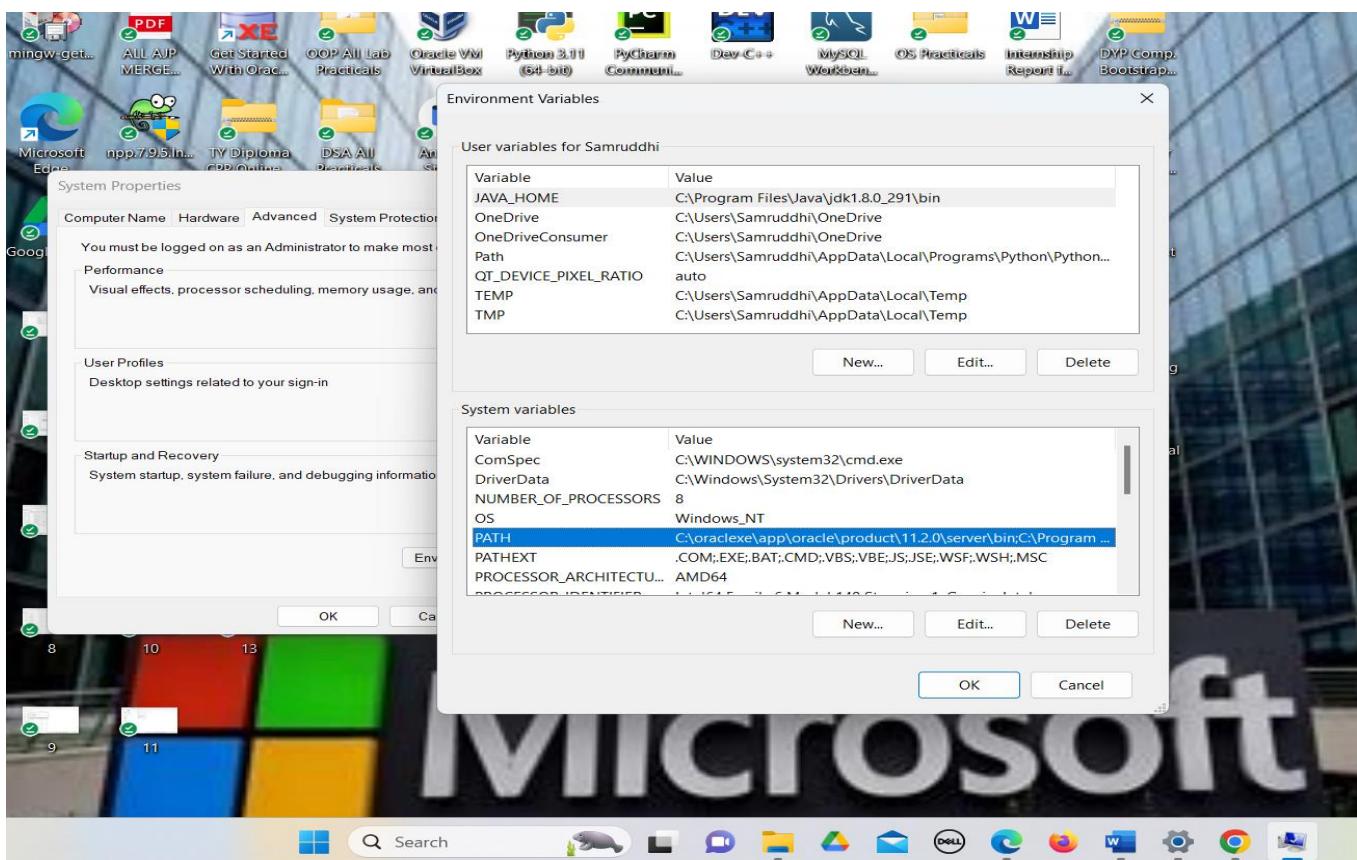
STEP 12:



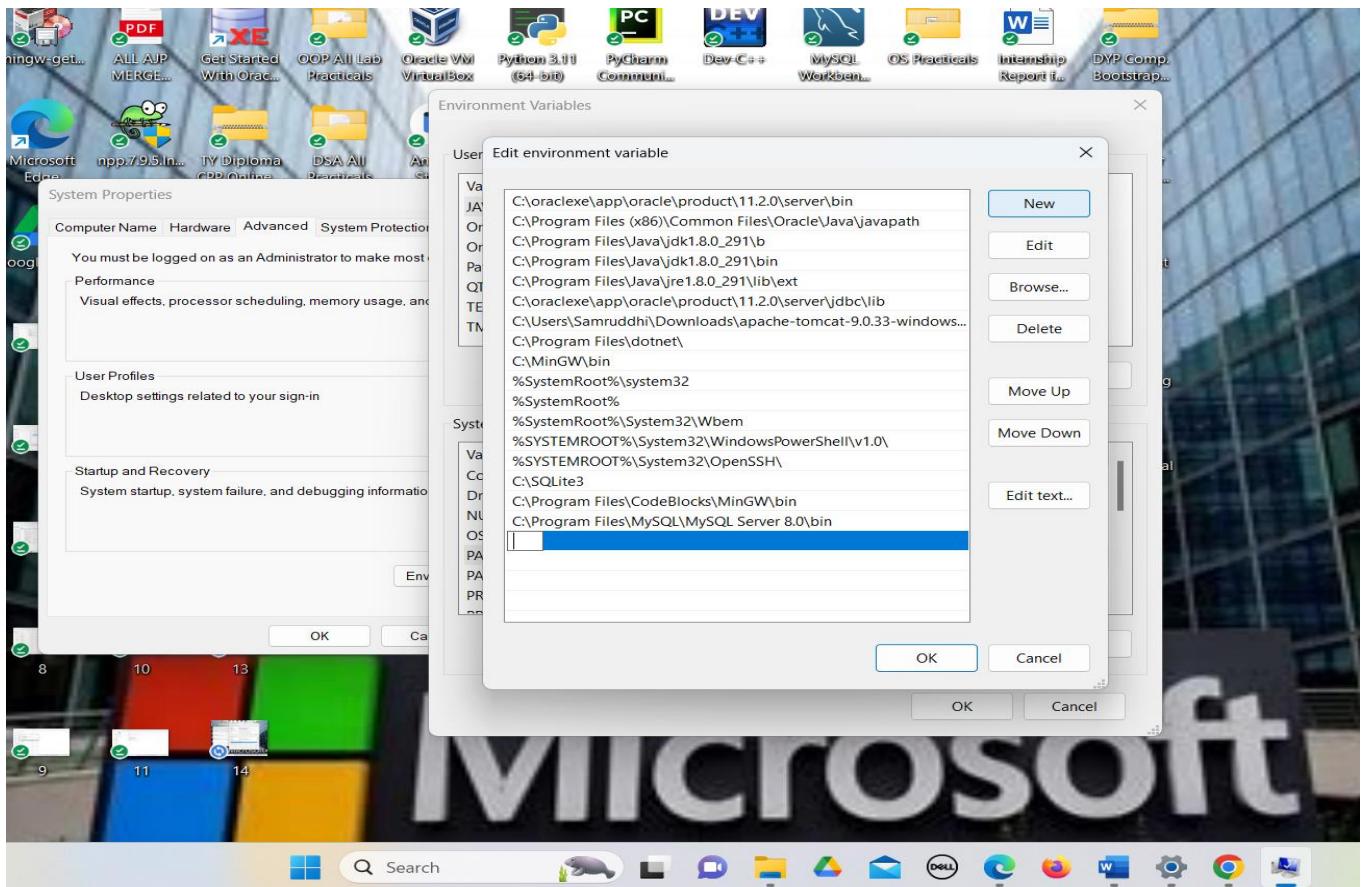
STEP 13:



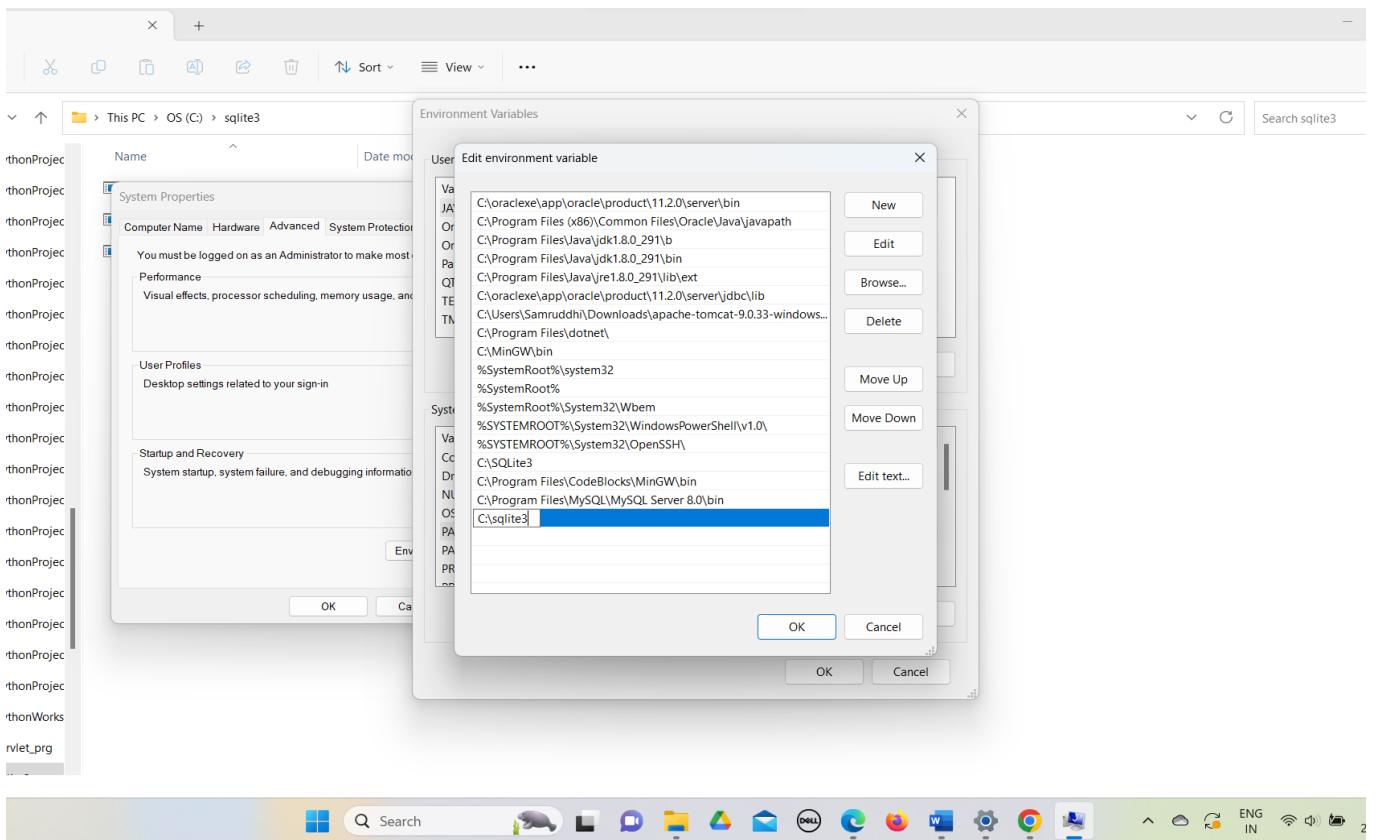
STEP 14:



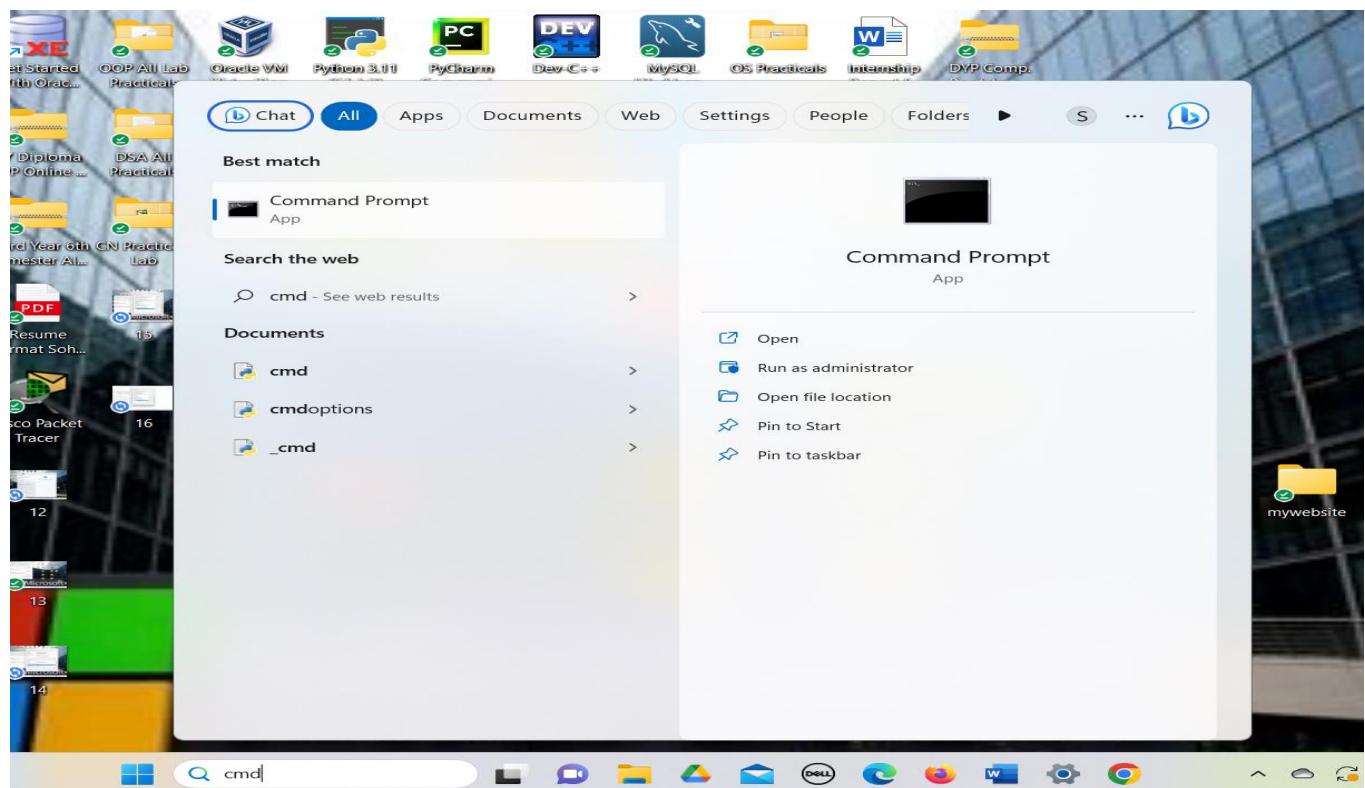
STEP 15:



STEP 16:



STEP 17:



STEP 18:

```
Administrator: Command Prompt - sqlite3
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>sqlite3
SQLite version 3.41.2 2023-03-22 11:56:21
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

STEP 19:

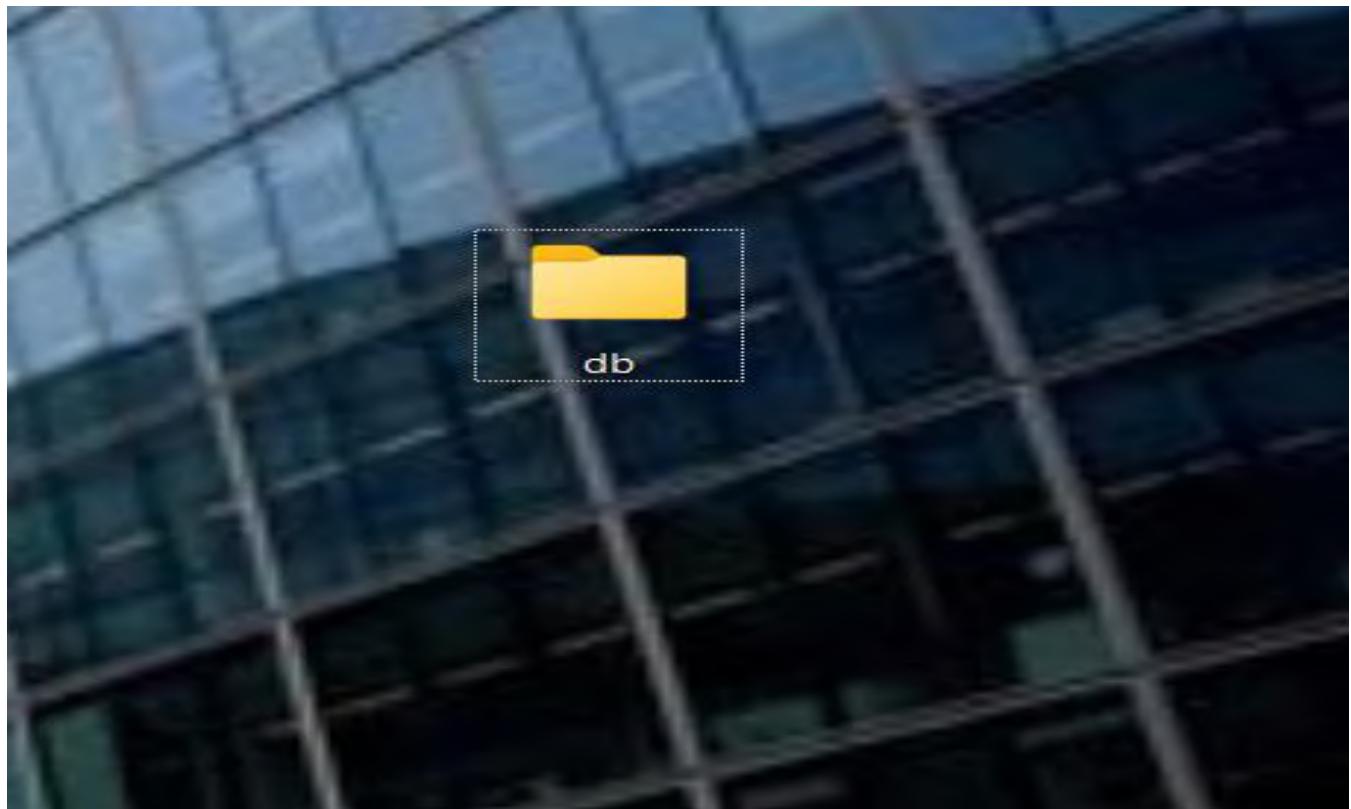
```
Administrator: Command Prompt - sqlite3
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>sqlite3
SQLite version 3.41.2 2023-03-22 11:56:21
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .help
.archive...          Manage SQL archives
.auth ON|OFF         Show authorizer callbacks
.backup ?DB? FILE   Backup DB (default "main") to FILE
.bail on|off         Stop after hitting an error. Default OFF
.binary on|off       Turn binary output on or off. Default OFF
.cd DIRECTORY       Change the working directory to DIRECTORY
.changes on|off      Show number of rows changed by SQL
.check GLOB         Fail if output since . testcase does not match
.clone NEWDB        Clone data into NEWDB from the existing database
.connection [close] [#] Open or close an auxiliary database connection
.databases           List names and files of attached databases
.dbconfig ?op? ?val? List or change sqlite3_db_config() options
.dbinfo ?DB?         Show status information about the database
.dump ?OBJECTS?     Render database content as SQL
.echo on|off         Turn command echo on or off
.eqp on|off|full|... Enable or disable automatic EXPLAIN QUERY PLAN
.excel               Display the output of next command in spreadsheet
.exit ?CODE?         Exit this program with return-code CODE
.expert              EXPERIMENTAL. Suggest indexes for queries
.explain ?on|off|auto? Change the EXPLAIN formatting mode. Default: auto
.filectrl CMD ...    Run various sqlite3_file_control() operations
.fullschema ?-indent? Show schema and the content of sqlite_stat tables
.headers on|off       Turn display of headers on or off
.help ?-all? ?PATTERN? Show help text for PATTERN
.import FILE TABLE   Import data from FILE into TABLE
.impostor INDEX TABLE Create impostor table TABLE on index INDEX
.indexes ?TABLE?     Show names of indexes
.limit ?LIMIT? ?VAL? Display or change the value of an SQLITE_LIMIT
.lint OPTIONS        Report potential schema issues.
.load FILE ?ENTRY?  Load an extension library
.log FILE|off        Turn logging on or off. FILE can be stderr/stdout
.mode MODE ?OPTIONS? Set output mode
.nonce STRING        Suspend safe mode for one command if nonce matches
.nullvalue STRING    Use STRING in place of NULL values
.once ?OPTIONS? ?FILE? Output for the next SQL command only to FILE
.open ?OPTIONS? ?FILE? Close existing database and reopen FILE
.output ?FILE?       Send output to FILE or stdout if FILE is omitted
.parameter CMD ...  Manage SQL parameter bindings
.print STRING...     Print literal STRING
.progress N          Invoke progress handler after every N opcodes
```

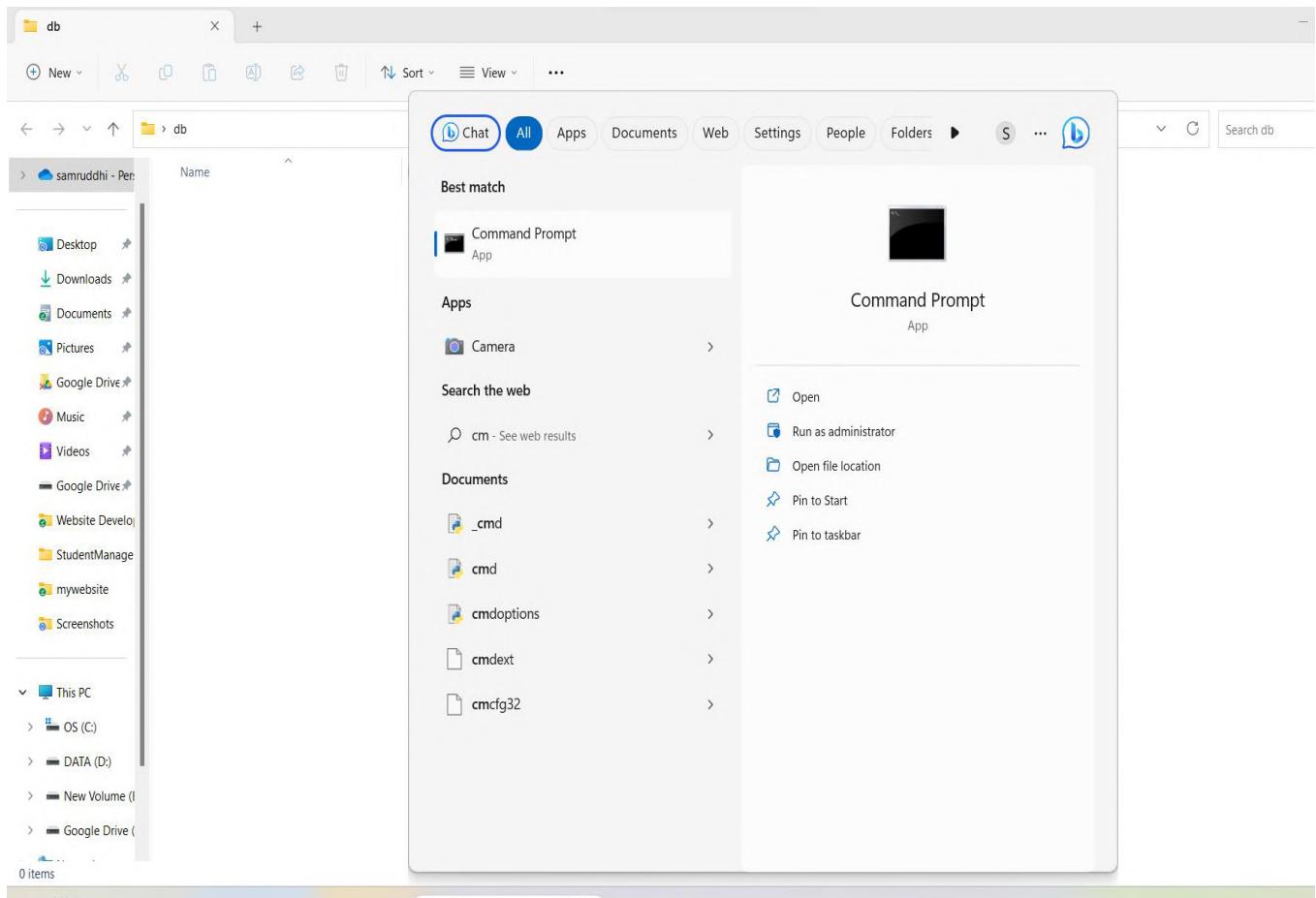
STEP 20:

```
Administrator: Command Prompt - sqlite3
.timeout MS          Try opening locked tables for MS milliseconds
.timer on|off         Turn SQL timer on or off
.trace ?OPTIONS?     Output each SQL statement as it is run
.version             Show source, library and compiler versions
.vfsinfo ?AUX?        Information about the top-level VFS
.vfslist              List all available VFSes
.vfsname ?AUX?        Print the name of the VFS stack
.width NUM1 NUM2 ...  Set minimum column widths for columnar output
sqlite> .quit
```

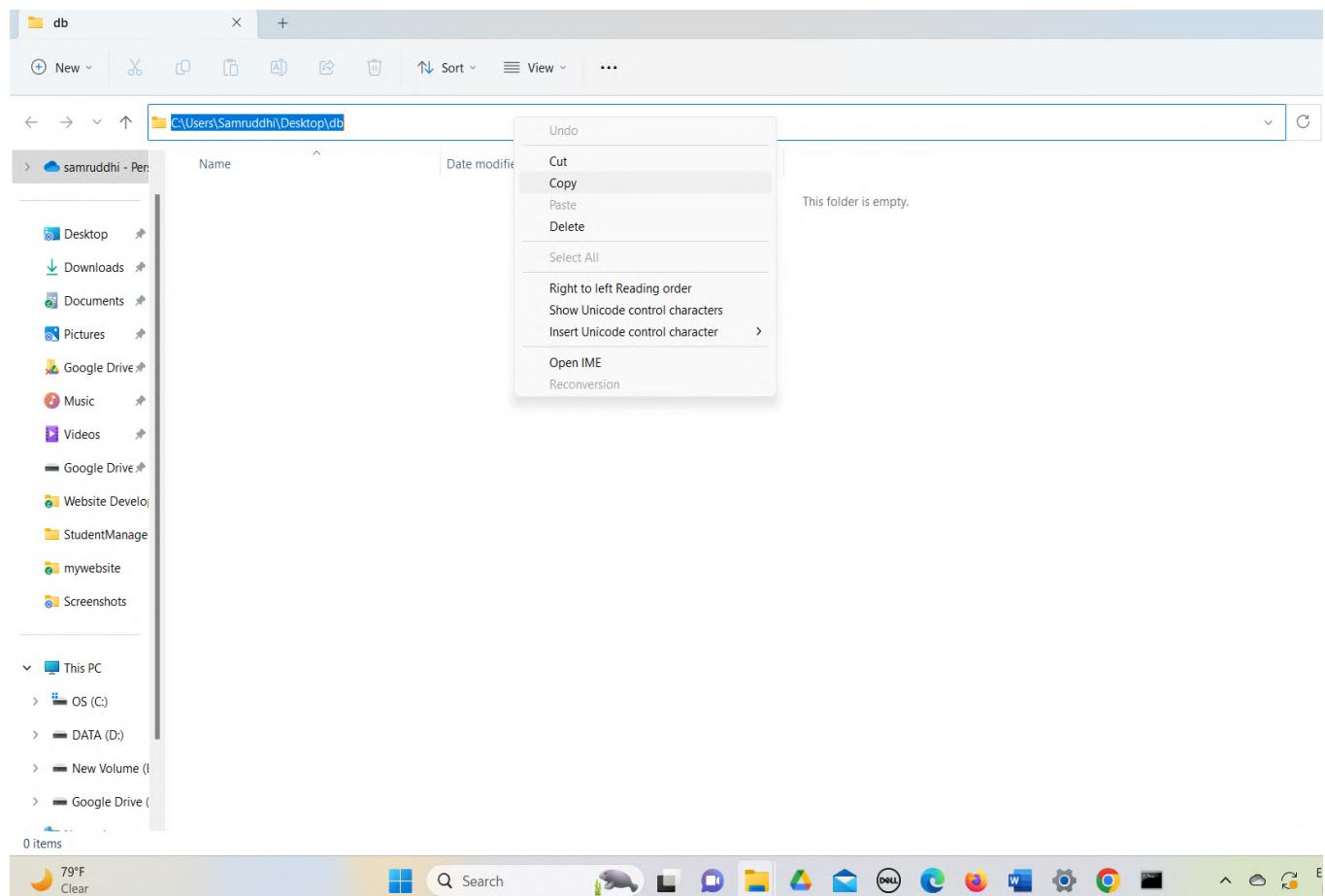
STEP 21:



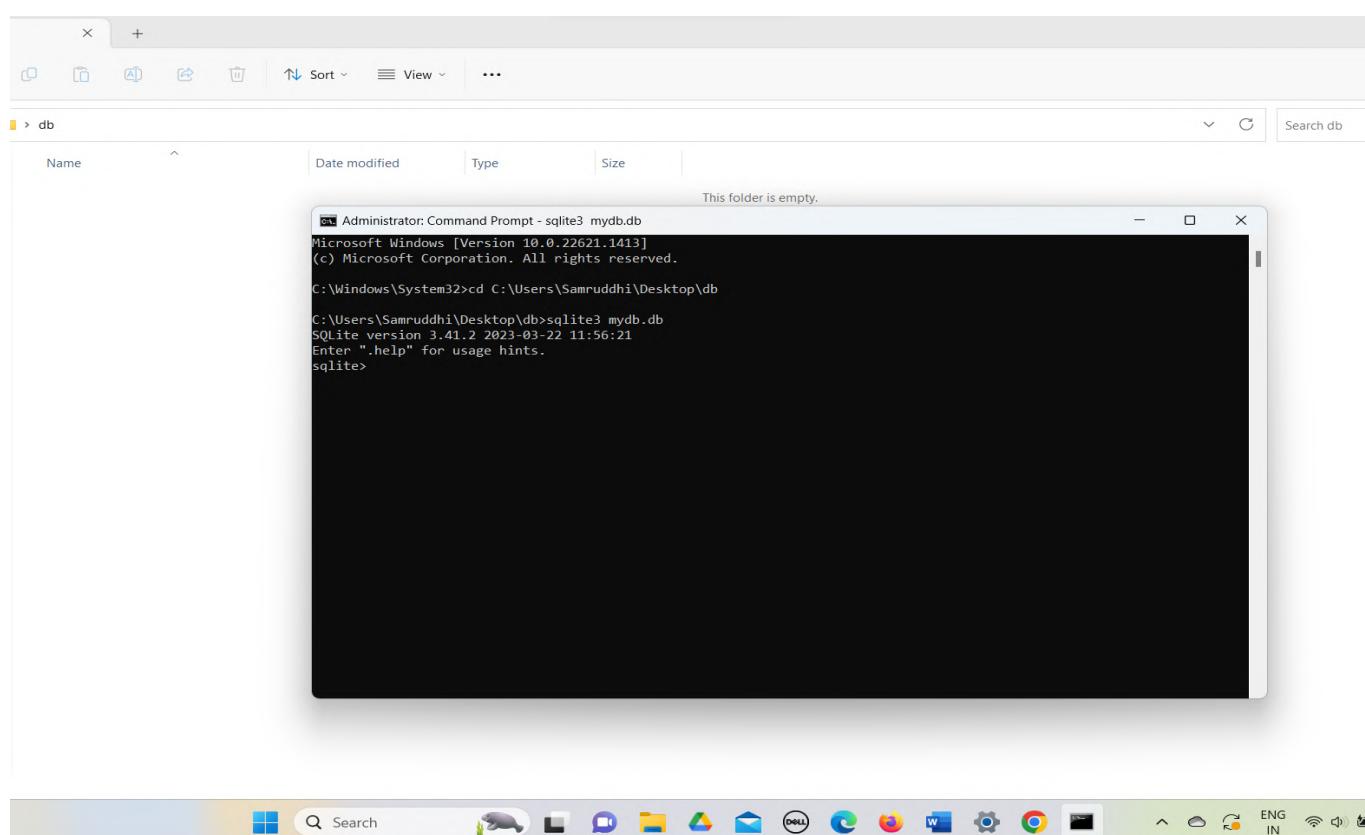
STEP 22:



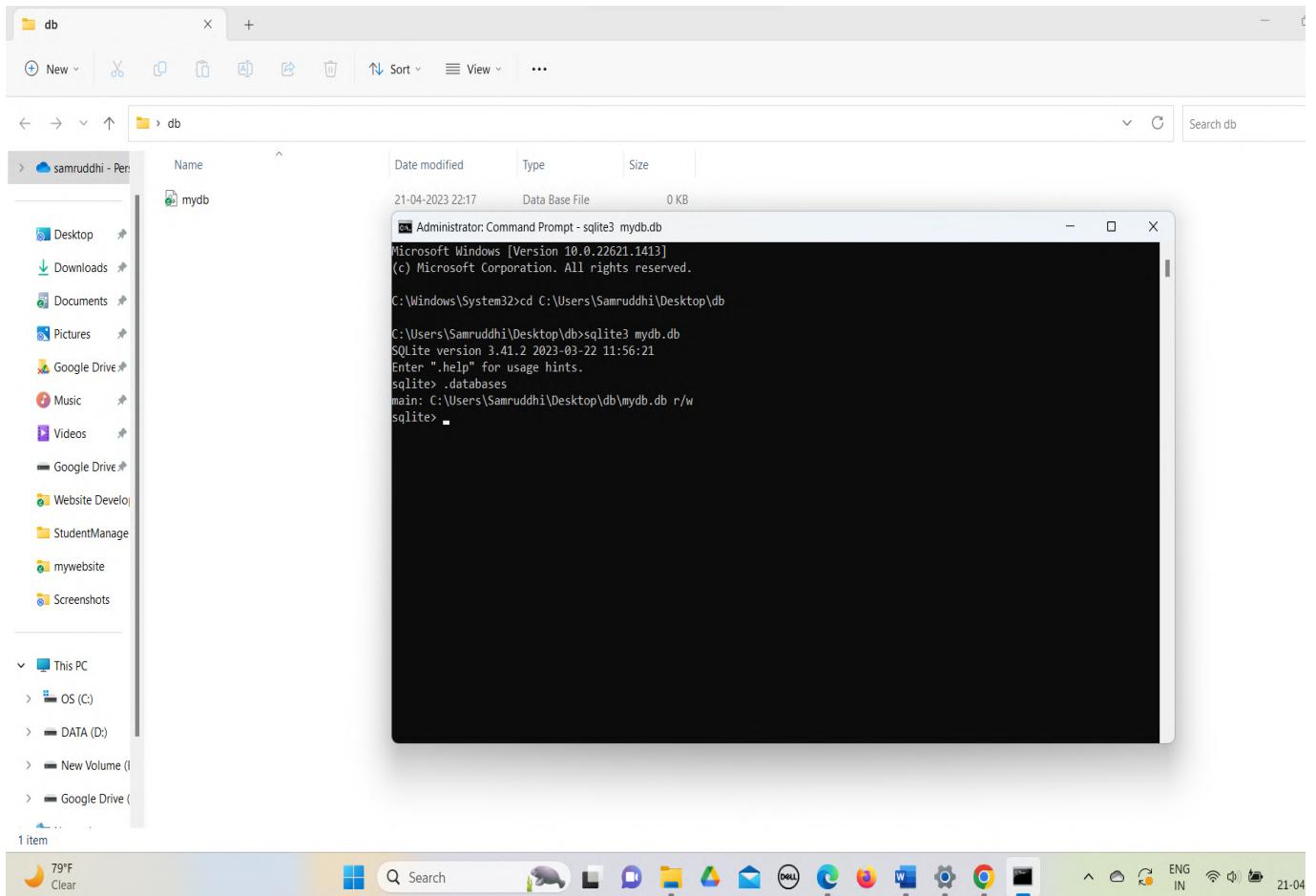
STEP 23:



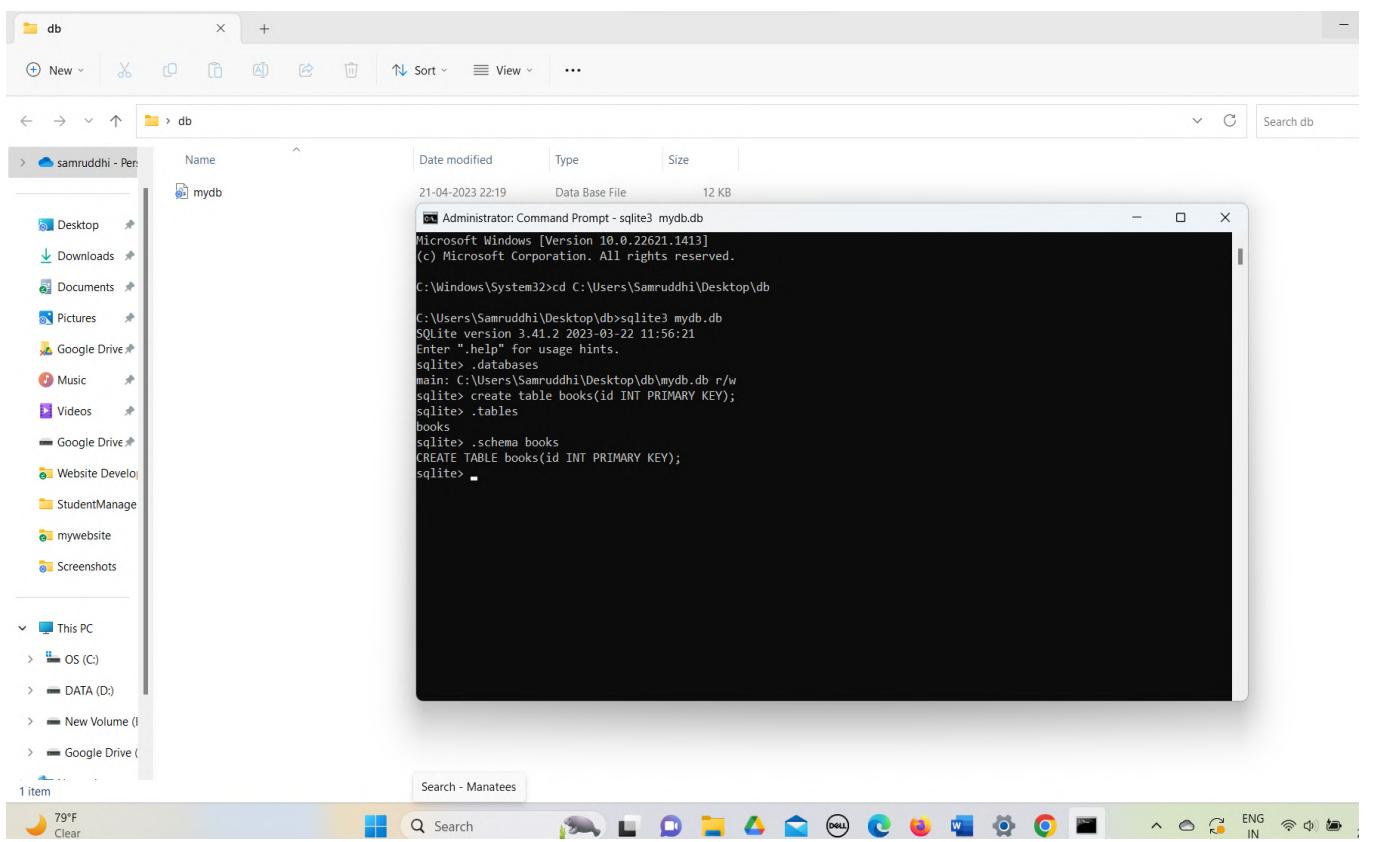
STEP 24:



STEP 25:

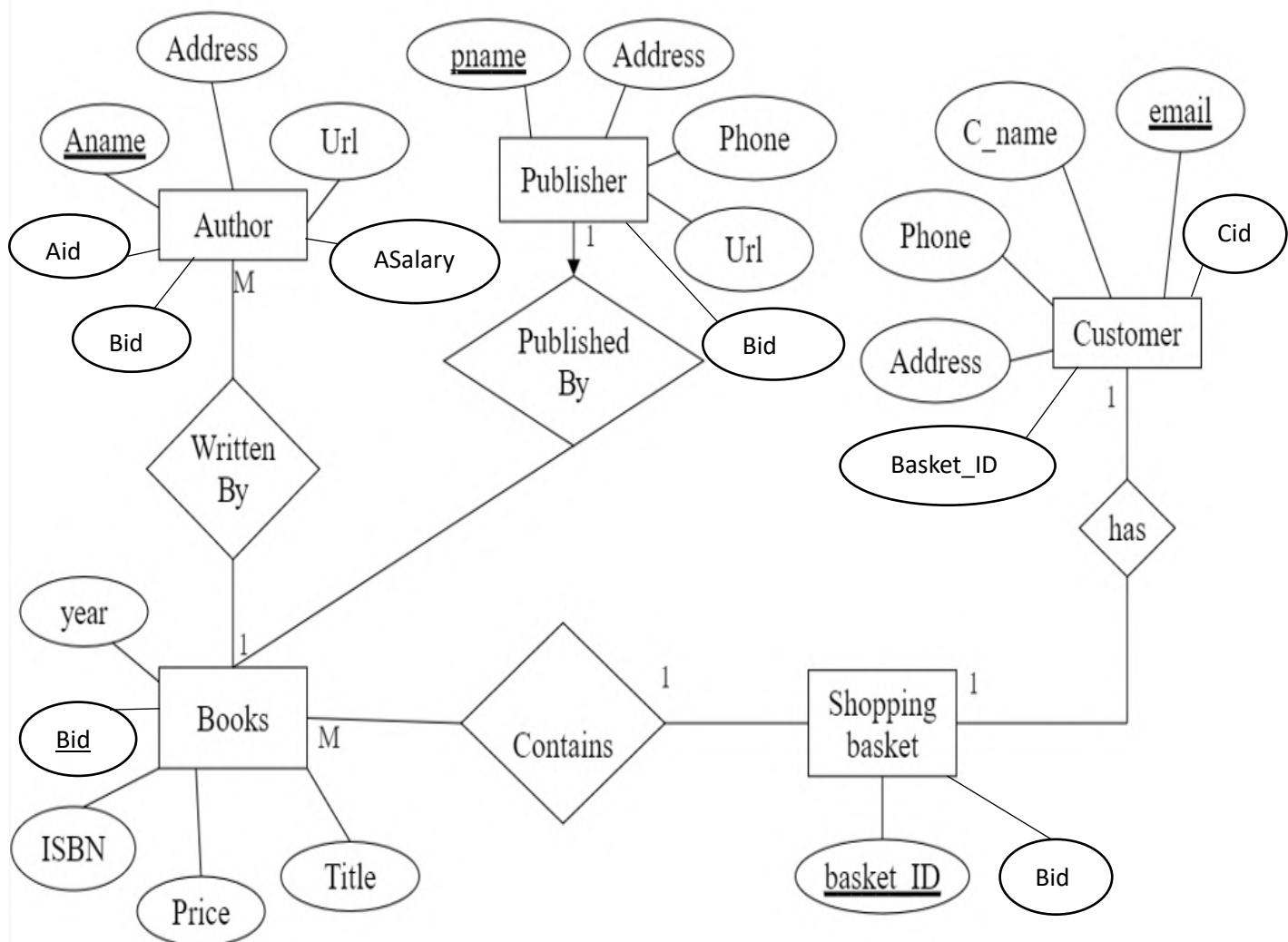


STEP 26:



Group B: MySQL

1. Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system.



Group B: MySQL

2.Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them.

Create Database:

```
create database college;
```

```
use college;
```

The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a code editor window titled "practical no. 1(create database...)". It contains the following SQL commands:

```
1 •  create database college;
2 •  use college;
3
4
```

In the bottom-right pane, there is an "Output" window titled "Action Output". It displays the results of the executed commands:

#	Time	Action	Message
1	11:22:59	create database college	1 row(s) affected
2	11:22:59	use college	0 row(s) affected

The bottom of the screen shows the Windows taskbar with various application icons.

Create table:

```
create table college_info(clg_id int ,college_name varchar(10),address varchar(10),
Branches varchar(10));

insert into college_info values(1,'GPA','Awasaki','IT');

insert into college_info values(2,'GPP','Pune','COMPUTER');

insert into college_info values(3,'GPM','Mumbai','MECHANICAL');

insert into college_info values(5,'GPN','Nashik','E&TC');

insert into college_info values(4,'GPK','Karad','ELECTRICAL');

select *from college_info;
```

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

- SQL Editor:** Displays the SQL code used to create the table and insert data.
- Result Grid:** Shows the resulting data from the SELECT query, which is empty.
- Action Output:** Shows the log of actions taken, including successful inserts and a failed select statement due to a table not existing.
- Toolbar:** Standard MySQL Workbench toolbar with various icons for database management.

clg_id	college_name	address	Branches
1	GPA	Awasaki	IT
2	GPP	Pune	COMPUTER
3	GPM	Mumbai	MECHANICAL
5	GPN	Nashik	E&TC
4	GPK	Karad	ELECTRICAL

Output:

#	Time	Action	Message
5	11:26:59	insert into college_info values(2,'GPP','Pune','COMPUTER')	1 row(s) affected
6	11:26:59	insert into college_info values(3,'GPM','Mumbai','MECHANICAL')	1 row(s) affected
7	11:26:59	insert into college_info values(5,'GPN','Nashik','E&TC')	1 row(s) affected
8	11:26:59	insert into college_info values(4,'GPK','Karad','ELECTRICAL')	1 row(s) affected
9	11:26:59	select 'from college LIMIT 0, 1000'	Error Code: 1146. Table 'college.college' doesn't exist
10	11:27:09	select 'from college_info LIMIT 0, 1000'	5 row(s) returned

Alter Command:

```
alter table college_info
```

```
add column(college_course varchar(10));
```

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

- SQL Editor:** Contains the following SQL code:

```
7 •  insert into college_info values(2,'GPP','Pune','COMPUTER');
8 •  insert into college_info values(3,'GPM','Mumbai','MECHANICAL');
9 •  insert into college_info values(5,'GPN','Nashik','E&TC');
10 • insert into college_info values(4,'GPK','Karad','ELECTRICAL');
11
12 • alter table college_info
13   add column(college_course varchar(10));
14
15
16 • select *from college_info;
```
- Result Grid:** Displays the data from the college_info table.

dg_id	college_name	address	Branches	college_course
1	GPA	Awasari	IT	NULL
2	GPP	Pune	COMPUTER	NULL
3	GPM	Mumbai	MECHANICAL	NULL
5	GPN	Nashik	E&TC	NULL
4	GPK	Karad	ELECTRICAL	NULL
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message
7	11:26:59	insert into college_info values(5,'GPN','Nashik','E&TC')	1 row(s) affected
8	11:26:59	insert into college_info values(4,'GPK','Karad','ELECTRICAL')	1 row(s) affected
9	11:26:59	select 'from college LIMIT 0, 1000'	Error Code: 1146. Table 'college.college' doesn't exist
10	11:27:09	select 'from college LIMIT 0, 1000'	5 row(s) returned
11	11:29:20	alter table college_info add column(college_course varchar(10))	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
12	11:29:24	select 'from college LIMIT 0, 1000'	5 row(s) returned

Rename command:

```
RENAME table college_info to college_details;
```

```
select *from college_details;
```

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a SQL editor window containing the following code:

```
7 • insert into college_info values(2,'GPP','Pune','COMPUTER');
8 • insert into college_info values(3,'GPM','Mumbai','MECHANICAL');
9 • insert into college_info values(5,'GPN','Nashik','E&TC');
10 • insert into college_info values(4,'GPK','Karad','ELECTRICAL');
11
12 • alter table college_info
13   add column(college_course varchar(10));
14
15 • RENAME table college_info to college_details;
16 • select *from college_details;
```

Below the SQL editor is a Result Grid window displaying the data from the 'college_details' table:

dg_id	college_name	address	Branches	college_course
1	GPA	Awasari	IT	NULL
2	GPP	Pune	COMPUTER	NULL
3	GPM	Mumbai	MECHANICAL	NULL
5	GPN	Nashik	E&TC	NULL
4	GPK	Karad	ELECTRICAL	NULL

At the bottom, the Output pane shows the results of the RENAME command and a SELECT query:

Action	Output
1 11:31:29 RENAME table college_info to college_details	Message 0 row(s) affected
2 11:31:36 select *from college_details LIMIT 0, 1000	Message 5 row(s) returned

Truncate Command:

truncate table college_info;

The screenshot shows the MySQL Workbench interface. In the top-left, the title bar says "MySQL Workbench" and "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. The left sidebar has a "Navigator" section with "SCHEMAS" and a "sys" entry. The main area contains a SQL editor tab titled "practical no.1(create database)..." with the following code:

```
12 • alter table college_info
13   add column(college_course varchar(10));
14
15 • RENAME table college_info to college_details;
16 • select *from college_details;
17 • select *from college_info;
18
19 • truncate table college_info;
20 • drop database college;
```

The SQL editor has a status bar at the bottom right with "Automatic context disabled. Use the to manually get help current caret position toggle automatic".

Below the SQL editor is an "Information" panel with tabs for "Administration" and "Schemas". It shows "No object selected".

At the bottom, there is a "college_info 5" tab showing the "Output" of the query execution. The output table has columns: #, Time, Action, Message, Duration / Fetch. The log entries are:

#	Time	Action	Message	Duration / Fetch
8	23:38:22	insert into college_info values(5,'GPN','Nashik','EATC')	1 row(s) affected	0.016 sec
9	23:38:22	insert into college_info values(4,'GPK','KaraD','ELECTRICAL')	1 row(s) affected	0.000 sec
10	23:38:28	alter table college_info add column(college_course varchar(10))	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
11	23:38:36	select *from college_info LIMIT 0, 50000	5 row(s) returned	0.000 sec / 0.0
12	23:38:45	truncate table college_info	0 row(s) affected	0.015 sec
13	23:38:52	select *from college_info LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.0

The system tray at the bottom shows weather (77°F), search, and other system icons. The date and time are 21-04-21.

Drop Command:

drop database college;

The screenshot shows a MySQL Workbench interface. The SQL tab contains the following code:

```
3
4 • create table college_info(clg_id int ,college_name varchar(10),address varchar(10),
5   Branches varchar(10));
6 • insert into college_info values(1,'GPA','Awasaki','IT');
7 • insert into college_info values(2,'GPP','Pune','COMPUTER');
8 • insert into college_info values(3,'GPM','Mumbai','MECHANICAL');
9 • insert into college_info values(5,'GPN','Nashik','E&TC');
10 • insert into college_info values(4,'GPK','Karad','ELECTRICAL');
11
12 • alter table college_info
13   add column(college_course varchar(10));
14
15 • RENAME table college_info to college_details;
16 • select *from college_details;
17 • select *from college_info;
18
19 • drop database college;
```

The Output tab shows the results of the actions:

#	Time	Action	Message
1	11:31:29	RENAME table college_info to college_details	0 row(s) affected
2	11:31:36	select *from college_details LIMIT 0, 1000	5 row(s) returned
3	11:33:07	drop database college	1 row(s) affected

DDL QUERIES:

```
create database college;
```

```
use college;
```

```
create table college_info(clg_id int ,college_name varchar(10),address varchar(10),
```

```
Branches varchar(10));
```

```
insert into college_info values(1,'GPA','Awasari','IT');
```

```
insert into college_info values(2,'GPP','Pune','COMPUTER');
```

```
insert into college_info values(3,'GPM','Mumbai','MECHANICAL');
```

```
insert into college_info values(5,'GPN','Nashik','E&TC');
```

```
insert into college_info values(4,'GPK','Karad','ELECTRICAL');
```

```
alter table college_info
```

```
add column(college_course varchar(10));
```

```
RENAME table college_info to college_details;
```

```
select *from college_details;
```

```
select *from college_info;
```

```
truncate table college_info;
```

```
drop database college;
```

Group B: MySQL

3. Create Table with primary key and foreign key constraints.

a. Alter table with add n modify

b. Drop table

Creating tables with primary key and foreign key constraints.

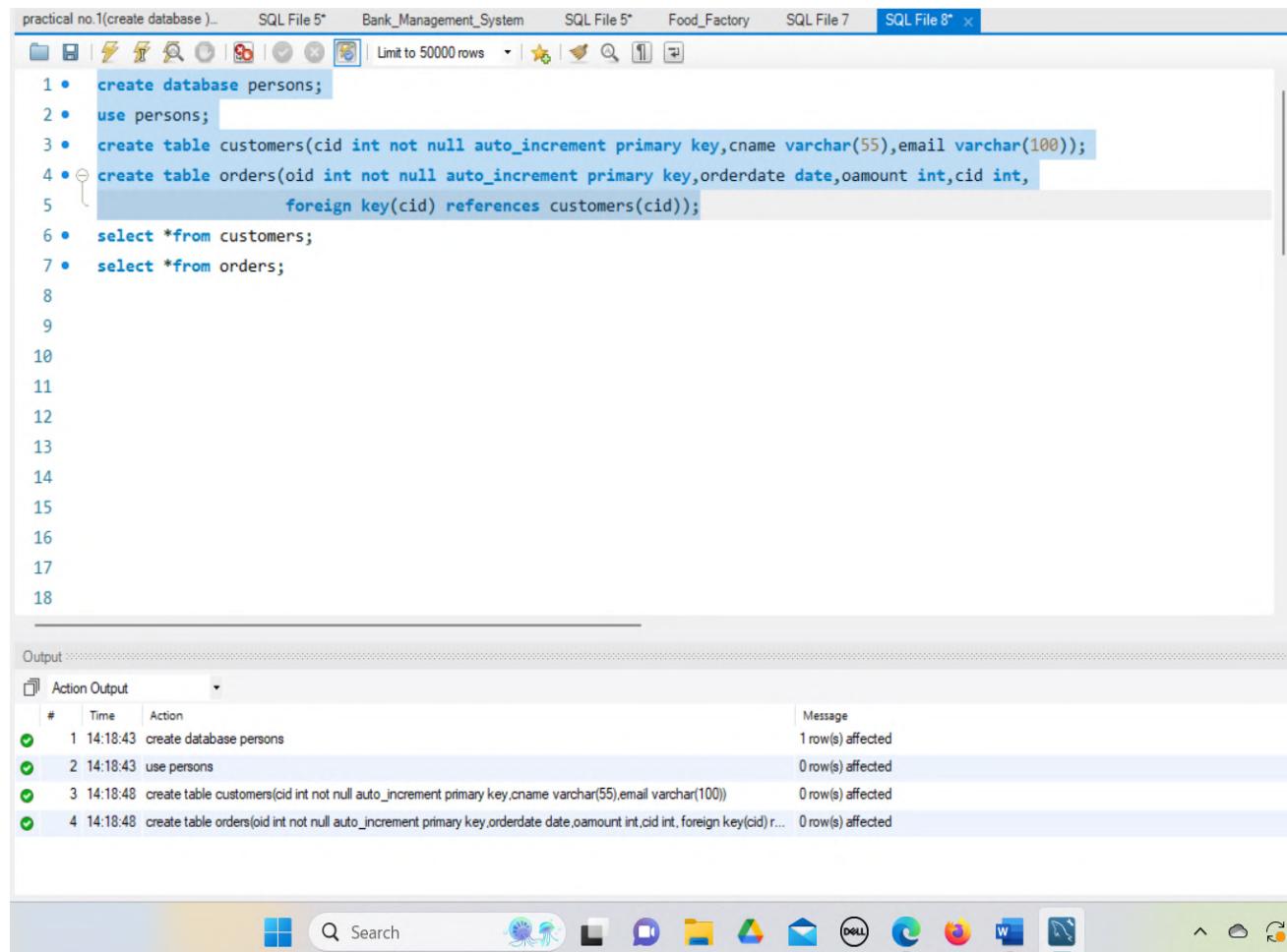
```
create database persons;
```

```
use persons;
```

```
create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
```

```
create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
```

```
foreign key(cid) references customers(cid));
```



```
practical no.1(create database ... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* ×
 1 • create database persons;
 2 • use persons;
 3 • create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
 4 • create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
   5   foreign key(cid) references customers(cid));
 6 • select *from customers;
 7 • select *from orders;
 8
 9
10
11
12
13
14
15
16
17
18
```

Output:

#	Time	Action	Message
1	14:18:43	create database persons	1 row(s) affected
2	14:18:43	use persons	0 row(s) affected
3	14:18:48	create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100))	0 row(s) affected
4	14:18:48	create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int, foreign key(cid) r...	0 row(s) affected

select *from customers;

The screenshot shows the SQL Server Management Studio interface. In the top ribbon, the tabs are practical no.1(create database ...), SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, and SQL File 8*. The SQL File 8* tab is active. The query window contains the following SQL code:

```
1 •  create database persons;
2 •  use persons;
3 •  create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
4 •  create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
5   foreign key(cid) references customers(cid));
6 •  select *from customers;
7 •  select *from orders;
8
9
10
11
12
```

The result grid shows the following data:

cid	cname	email
*	HULL	HULL

The status bar at the bottom indicates "customers 31 x". Below the query window, the Action Output pane shows the following log entries:

#	Time	Action	Message
1	14:18:43	create database persons	1 row(s) affected
2	14:18:43	use persons	0 row(s) affected
3	14:18:48	create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100))	0 row(s) affected
4	14:18:48	create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int, foreign key(cid) ...)	0 row(s) affected
5	14:19:43	select *from customers LIMIT 0, 50000	0 row(s) returned

The taskbar at the bottom of the screen includes icons for File, Edit, View, Tools, Options, Help, and several system icons.

select *from orders;

The screenshot shows the SQL Server Management Studio interface. In the top ribbon, the tabs are practical no.1(create database ...), SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, and SQL File 8*. The SQL File 8* tab is active. The query window contains the following SQL code:

```
1 •  create database persons;
2 •  use persons;
3 •  create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
4 •  create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
5   foreign key(cid) references customers(cid));
6 •  select *from customers;
7 •  select *from orders;
8
9
10
11
12
```

The result grid shows the following data:

oid	orderdate	oamount	cid
*	HULL	HULL	HULL

The status bar at the bottom indicates "orders 32 x". Below the query window, the Action Output pane shows the following log entries:

#	Time	Action	Message
1	14:18:43	create database persons	1 row(s) affected
2	14:18:43	use persons	0 row(s) affected
3	14:18:48	create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100))	0 row(s) affected
4	14:18:48	create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int, foreign key(cid) ...)	0 row(s) affected
5	14:19:43	select *from customers LIMIT 0, 50000	0 row(s) returned
6	14:20:40	select *from orders LIMIT 0, 50000	0 row(s) returned

The taskbar at the bottom of the screen includes icons for File, Edit, View, Tools, Options, Help, and several system icons.

a. Alter table with add n modify

add constraints customer table:

```
create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
```

```
insert into customers values(1,'Samruddhi','samruddhi@gmail.com');
```

```
insert into customers values(2,'Aditi','aditi@gmail.com');
```

```
insert into customers values(3,'Soham','soham@gmail.com');
```

```
select *from customers;
```

```
ALTER TABLE customers add salary int;
```

```
describe customers;
```

The screenshot shows the MySQL Workbench interface with several tabs at the top: practical no.1(create database ...), SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, and SQL File 8* (active). Below the tabs is a toolbar with various icons. The main area contains a code editor with numbered lines 17 through 25. Lines 18-24 show the creation of the 'customers' table, insertion of three rows, selection of all columns, alteration of the table to add a 'salary' column, and description of the table. Lines 26 and 27 are blank. The bottom section shows the 'Result Grid' with the data from the 'customers' table:

	cid	cname	email	salary
▶	1	Samruddhi	samruddhi@gmail.com	HULL
▶	2	Aditi	aditi@gmail.com	HULL
▶	3	Soham	soham@gmail.com	HULL
*	HULL	HULL	HULL	HULL

The 'Output' pane below shows the execution history with actions 2 through 7 and their corresponding messages. The system tray at the bottom includes icons for search, file, messaging, folder, Google Drive, email, Dell, Firefox, Word, and others.

```
17
18 • create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
19 • insert into customers values(1,'Samruddhi','samruddhi@gmail.com');
20 • insert into customers values(2,'Aditi','aditi@gmail.com');
21 • insert into customers values(3,'Soham','soham@gmail.com');
22 • select *from customers;
23 • ALTER TABLE customers add salary int;
24 • describe customers;
25
26
27
28
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	cid	cname	email	salary
▶	1	Samruddhi	samruddhi@gmail.com	HULL
▶	2	Aditi	aditi@gmail.com	HULL
▶	3	Soham	soham@gmail.com	HULL
*	HULL	HULL	HULL	HULL

customers 35 x Apply

Output

Action Output

#	Time	Action	Message
2	14:32:02	insert into customers values(2,'Aditi','aditi@gmail.com')	1 row(s) affected
3	14:32:02	insert into customers values(3,'Soham','soham@gmail.com')	1 row(s) affected
4	14:32:02	select *from customers LIMIT 0, 50000	3 row(s) returned
5	14:32:18	ALTER TABLE customers add salary int	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
6	14:32:22	describe customers	4 row(s) returned
7	14:32:33	select *from customers LIMIT 0, 50000	3 row(s) returned

add constraints order table:

```
create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
foreign key(cid) references customers(cid));
```

```
insert into orders values(1,'2023/2/3',55,1);
```

```
insert into orders values(2,'2022/7/10',87,2);
```

```
insert into orders values(3,'2020/6/8',76,3);
```

```
select *from orders;
```

```
ALTER TABLE orders add oname varchar(20);
```

```
describe orders;
```

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

- SQL Editor:** Contains the SQL code for creating the 'orders' table, inserting three rows, selecting from the table, altering it to add a column 'oname', and describing the table.
- Result Grid:** Displays the structure of the 'orders' table with columns: Field, Type, Null, Key, Default, Extra. The table has 5 columns: oid (int, NO, PRI, auto_increment), orderdate (date, YES), oamount (int, YES), cid (int, YES, MUL), and oname (varchar(20), YES).
- Action Output:** Shows the log of actions with their times, descriptions, and messages. It includes insertions of rows 1, 2, and 3, a select query, an alter table command, and a describe command. The log also indicates 1 row(s) affected for each insert, 3 row(s) returned for the select, and 0 row(s) affected for the alter table.

modify constraints customer table:

```
create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));  
insert into customers values(1,'Samruddhi','samruddhi@gmail.com');  
insert into customers values(2,'Aditi','aditi@gmail.com');  
insert into customers values(3,'Soham','soham@gmail.com');  
select *from customers;  
ALTER TABLE customers modify cname varchar(20);  
describe customers;
```

The screenshot shows the MySQL Workbench interface with several tabs at the top: practical no.1(create database ...), SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, and SQL File 8*. The SQL File 8* tab contains the executed SQL code. Below the tabs is a toolbar with various icons. The main area displays the results of the executed queries. A 'Result Grid' table shows the structure of the 'customers' table:

Field	Type	Null	Key	Default	Extra
cid	int	NO	PRI	NULL	auto_increment
cname	varchar(20)	YES		NULL	
email	varchar(100)	YES		NULL	

Below the table is a 'Result 39' grid showing the history of actions and their messages:

#	Time	Action	Message
4	15:00:23	insert into customers values(1,'Samruddhi','samruddhi@gmail.com')	1 row(s) affected
5	15:00:23	insert into customers values(2,'Aditi','aditi@gmail.com')	1 row(s) affected
6	15:00:23	insert into customers values(3,'Soham','soham@gmail.com')	1 row(s) affected
7	15:00:23	select *from customers LIMIT 0, 50000	3 row(s) returned
8	15:00:28	ALTER TABLE customers modify cname varchar(20)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
9	15:00:32	describe customers	3 row(s) returned

The bottom of the interface features a toolbar with various system icons.

modify constraints order table:

```
create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
foreign key(cid) references customers(cid));
```

```
insert into orders values(1,'2023/2/3',55,1);
```

```
insert into orders values(2,'2022/7/10',87,2);
```

```
insert into orders values(3,'2020/6/8',76,3);
```

```
select *from orders;
```

```
ALTER TABLE orders modify oamount double;
```

```
describe orders;
```

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

- SQL Editor:** Contains the SQL code for creating the 'orders' table, inserting three rows, modifying the 'oamount' column, and describing the table.
- Result Grid:** Displays the structure of the 'orders' table with columns: Field, Type, Null, Key, Default, Extra. The table has four rows: oid (int, NO, PRI, auto_increment), orderdate (date, YES, NULL), oamount (double, YES, NULL), and cid (int, YES, MUL, NULL).
- Action Output:** Shows the log of actions with their times and messages:
 - Row 2: insert into orders values(1,'2023/2/3',55,1) - 1 row(s) affected, 1 warning(s): 4095 Delimiter '/' in position 4 in datetime value '2023/2/3' at
 - Row 3: insert into orders values(2,'2022/7/10',87,2) - 1 row(s) affected, 1 warning(s): 4095 Delimiter '/' in position 4 in datetime value '2022/7/10' a
 - Row 4: insert into orders values(3,'2020/6/8',76,3) - 1 row(s) affected, 1 warning(s): 4095 Delimiter '/' in position 4 in datetime value '2020/6/8' at
 - Row 5: select *from orders LIMIT 0, 50000 - 3 row(s) returned
 - Row 6: ALTER TABLE orders modify oamount double - 3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
 - Row 7: describe orders - 4 row(s) returned

b. Drop table

ALTER TABLE customers drop email;

describe customers;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes tabs for 'practical no.1(create database)...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', and 'SQL File 8*'. The main query window contains the following SQL code:

```
3
4
5 •  create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
6 •  insert into customers values(1,'Samruddhi','samruddhi@gmail.com');
7 •  insert into customers values(2,'Aditi','aditi@gmail.com');
8 •  insert into customers values(3,'Soham','soham@gmail.com');
9 •  select *from customers;
10 • ALTER TABLE customers drop email;
11 • describe customers;
12
13
```

Below the code, the 'Result Grid' shows the structure of the 'customers' table:

Field	Type	Null	Key	Default	Extra
cid	int	NO	PRI	NULL	auto_increment
cname	varchar(20)	YES		NULL	

The 'Output' pane at the bottom displays the execution log:

#	Time	Action	Message
1	15:17:14	ALTER TABLE customers drop email	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	15:17:25	describe customers	2 row(s) returned

ALTER TABLE orders drop oamount;

describe orders;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes tabs for 'practical no.1(create database)...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', and 'SQL File 8*'. The main query window contains the following SQL code:

```
13
14 •  create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
15          foreign key(cid) references customers(cid));
16 •  insert into orders values(1,'2023/2/3',55,1);
17 •  insert into orders values(2,'2022/7/10',87,2);
18 •  insert into orders values(3,'2020/6/8',76,3);
19 •  select *from orders;
20 • ALTER TABLE orders drop oamount;
21 • describe orders;
22
23
```

Below the code, the 'Result Grid' shows the structure of the 'orders' table:

Field	Type	Null	Key	Default	Extra
oid	int	NO	PRI	NULL	auto_increment
orderdate	date	YES		NULL	
cid	int	YES	MUL	NULL	

The 'Output' pane at the bottom displays the execution log:

#	Time	Action	Message
1	15:17:14	ALTER TABLE customers drop email	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	15:17:25	describe customers	2 row(s) returned
3	15:24:08	ALTER TABLE orders drop oamount	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
4	15:24:08	describe orders	3 row(s) returned

```
drop table customers;
```

```
drop table orders;
```

```
drop database persons;
```

The screenshot shows a SQL development environment with multiple tabs at the top: 'practical no.1(create database ...)', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', and 'SQL File 8*'. The 'SQL File 8*' tab is active and contains the following SQL code:

```
11 • describe customers;
12
13
14 • create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
   foreign key(cid) references customers(cid));
15
16 • insert into orders values(1,'2023/2/3',55,1);
17 • insert into orders values(2,'2022/7/10',87,2);
18 • insert into orders values(3,'2020/6/8',76,3);
19 • select *from orders;
20 • ALTER TABLE orders drop oamount;
21 • describe orders;
22
23
24 • select *from customers,orders;
25 • drop table customers;
26 • drop table orders;
27 • drop database persons;
```

Below the code editor is an 'Output' panel titled 'Action Output' which displays the results of the executed command:

#	Time	Action	Message
1	15:28:05	drop database persons	2 row(s) affected

The system tray at the bottom of the screen shows various icons for system functions like search, file explorer, and network.

Create Table with primary key and foreign key constraints.

Alter table with add n modify

Drop table

```
create database persons;
use persons;
create table customers(cid int not null auto_increment primary key,cname varchar(55),email varchar(100));
create table orders(oid int not null auto_increment primary key,orderdate date,oamount int,cid int,
foreign key(cid) references customers(cid));

insert into customers values(1,'Samruddhi','samruddhi@gmail.com');
insert into customers values(2,'Aditi','aditi@gmail.com');
insert into customers values(3,'Soham','soham@gmail.com');
select *from customers;
ALTER TABLE customers add salary int;
ALTER TABLE customers modify cname varchar(20);
ALTER TABLE customers drop salary;
describe customers;

insert into orders values(1,'2023/2/3',55,1);
insert into orders values(2,'2022/7/10',87,2);
insert into orders values(3,'2020/6/8',76,3);
select *from orders;
ALTER TABLE orders add oname varchar(20);
ALTER TABLE orders modify oamount double;
ALTER TABLE orders drop oname;
describe orders;

select *from customers,orders;
drop table customers;
drop table orders;
drop database persons;
```

Group B: MySQL

4. Perform following SQL queries on the database created in assignment 1.

- Implementation of relational operators in SQL
- Boolean operators and pattern matching
- Arithmetic operations and built in functions
- Group functions
- Processing Date and Time functions
- Complex queries and set operators

```
create database Books;
```

```
use Books;
```

```
create table Books(Bid int primary key,year double,ISBN int,Price double,Title varchar(20));
```

```
insert into Books values(1001,2002,765,2200,'Advanced Java');
```

```
insert into Books values(2002,2011,987,1770,'Postgree SQL');
```

```
insert into Books values(3003,2021,324,1000,'Software Agile');
```

```
select *from Books;
```

```
create table Author(Aname varchar(20) primary key,Aid int,Address varchar(20),Url varchar(30),Asalary double,Bid int, foreign key(Bid) references Books(Bid));
```

```
insert into Author values('Albert',101,'Bengalore','Rhickey.com',267654,1001);
```

```
insert into Author values('Stephen King',202,'Kanada','Stephenking.com',198767,2002);
```

```
insert into Author values('DrPhillMcCraw',303,'USA','drphillmccraw.com',998642,3003);
```

```
select *from Author;
```

```
create table Publisher(pname varchar(20) primary key,Address varchar(20),PPhone varchar(10),Url varchar(30),Bid int,foreign key(Bid) references Books(Bid));
```

```
insert into Publisher values('John','Pune','9090987544','john.com',1001);
```

```
insert into Publisher values('Albert','mumbai','9865784321','albert.com',2002);
```

```
insert into Publisher values('hickey','kolkata','7689543221','hickey.com',3003);
```

```
select *from Publisher;
```

```
create table ShoppingBasket(basket_ID int primary key,Bid int,foreign key(Bid) references Books(Bid));
```

```
insert into ShoppingBasket values(298,1001);
```

```
insert into ShoppingBasket values(898,2002);
```

```
insert into ShoppingBasket values(760,3003);
```

```
select *from ShoppingBasket;
```

```
create table Customer(email varchar(30) primary key,Cid int,C_name varchar(20),CPhone varchar(10),Address varchar(30),CDOB date,basket_ID int,foreign key(basket_ID) references ShoppingBasket(basket_ID));
```

```
insert into Customer values('samruddhikangude@gmail.com',100,'samruddhi','7218380003','Pune','2017-06-06',298);
```

```
insert into Customer values('aditimagar@gmail.com',200,'aditi','8787878787','osmanabad','2015-05-04',898);
```

```
insert into Customer values('sohampawar@gmail.com',300,'soham','7765432234','Baramati','2019-09-12',760); select *from Customer; drop database Books;
```

Books Table:

The screenshot shows the SQL Server Management Studio interface. In the top ribbon, tabs include 'base ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (which is selected), and 'SQL File 10*'. Below the tabs is a toolbar with icons for file operations like New, Open, Save, and Print, along with search and filter tools. A status bar at the bottom indicates 'Limit to 50000 rows'.

The main area displays the following SQL code:

```
1 • create database Books;
2 • use Books;
3 • create table Books(Bid int primary key,year double,ISBN int,Price double,Title varchar(20));
4 • insert into Books values(1001,2002,765,2200,'Advanced Java');
5 • insert into Books values(2002,2011,987,1770,'PostgreSQL');
6 • insert into Books values(3003,2021,324,1000,'Software Agile');
7 • select *from Books;
8
9 • create table Author(Aname varchar(20) primary key,Aid int,Address varchar(20),Url varchar(30),Asalary double,Bid int,
10      foreign key(Bid) references Books(Bid));
```

Below the code is a 'Result Grid' table with columns: Bid, year, ISBN, Price, and Title. The data is as follows:

Bid	year	ISBN	Price	Title
1001	2002	765	2200	Advanced Java
2002	2011	987	1770	PostgreSQL
3003	2021	324	1000	Software Agile
*	HULL	HULL	HULL	HULL

At the bottom of the interface is a Windows taskbar with icons for various applications like File Explorer, Google Chrome, and Microsoft Word.

Author Table:

The screenshot shows the SQL Server Management Studio interface. The top ribbon and toolbar are identical to the previous screenshot. The main area displays the following SQL code:

```
8
9 • create table Author(Aname varchar(20) primary key,Aid int,Address varchar(20),Url varchar(30),Asalary double,Bid int,
10      foreign key(Bid) references Books(Bid));
11 • insert into Author values('Albert',101,'Bengaluru','R hickey.com',267654,1001);
12 • insert into Author values('Stephen King',202,'Kanada','Stephenking.com',198767,2002);
13 • insert into Author values('DrPhillMcCraw',303,'USA','drphillmccraw.com',998642,3003);
14 • select *from Author;
15
16 • create table Publisher(pname varchar(20) primary key,Address varchar(20),PPhone varchar(10),Url varchar(30),Bid int,
17      foreign key(Bid) references Books(Bid));
```

Below the code is a 'Result Grid' table with columns: Bid, year, ISBN, Price, and Title. The data is as follows:

Bid	year	ISBN	Price	Title
1001	2002	765	2200	Advanced Java
2002	2011	987	1770	PostgreSQL
3003	2021	324	1000	Software Agile
*	HULL	HULL	HULL	HULL

At the bottom of the interface is a Windows taskbar with icons for various applications like File Explorer, Google Chrome, and Microsoft Word.

Publisher Table:

SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*

```

14 • select *from Author;
15
16 • create table Publisher(pname varchar(20) primary key,Address varchar(20),PPhone varchar(10),Url varchar(30),Bid int,
17     foreign key(Bid) references Books(Bid));
18 • insert into Publisher values('John','Pune','9090987544','john.com',1001);
19 • insert into Publisher values('Albert','mumbai','9865784321','albert.com',2002);
20 • insert into Publisher values('hichey','kolkata','7689543221','hichey.com',3003);
21 • select *from Publisher;
22
23 • create table ShoppingBasket(basket_ID int primary key,Bid int.

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

pname	Address	PPhone	Url	Bid
Albert	mumbai	9865784321	albert.com	2002
hichey	kolkata	7689543221	hichey.com	3003
John	Pune	9090987544	john.com	1001
*	NULL	NULL	NULL	NULL

Publisher 222 x

Output

Action Output

#	Time	Action	Message
7	14:31:05	select 'from Books LIMIT 0, 50000	3 row(s) returned
8	14:34:12	create table Publisher(pname varchar(20) primary key,Address varchar(20),PPhone varchar(10),Url varchar(30),Bid int,foreign key(Bid) references Books(Bid));	0 row(s) affected
9	14:34:12	insert into Publisher values('John','Pune','9090987544','john.com',1001)	1 row(s) affected
10	14:34:12	insert into Publisher values('Albert','mumbai','9865784321','albert.com',2002)	1 row(s) affected
11	14:34:12	insert into Publisher values('hichey','kolkata','7689543221','hichey.com',3003)	1 row(s) affected
12	14:34:12	select 'from Publisher LIMIT 0, 50000	3 row(s) returned

Search

ShoppingBasket Table:

SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*

```

21 • select *from Publisher;
22
23 • create table ShoppingBasket(basket_ID int primary key,Bid int,
24     foreign key(Bid) references Books(Bid));
25 • insert into ShoppingBasket values(298,1001);
26 • insert into ShoppingBasket values(898,2002);
27 • insert into ShoppingBasket values(760,3003);
28 • select *from ShoppingBasket;
29
30 • create table Customer(email varchar(30) primary key,Cid int,C name varchar(20),CPhone varchar(10),Address varchar(30).

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

basket_ID	Bid
298	1001
898	2002
760	3003
*	NULL

ShoppingBasket 223 x

Output

Action Output

#	Time	Action	Message
12	14:34:12	select 'from Publisher LIMIT 0, 50000	3 row(s) returned
13	14:34:54	create table ShoppingBasket(basket_ID int primary key,Bid int,foreign key(Bid) references Books(Bid));	0 row(s) affected
14	14:34:54	insert into ShoppingBasket values(298,1001)	1 row(s) affected
15	14:34:54	insert into ShoppingBasket values(898,2002)	1 row(s) affected
16	14:34:54	insert into ShoppingBasket values(760,3003)	1 row(s) affected
17	14:34:54	select 'from ShoppingBasket LIMIT 0, 50000	3 row(s) returned

Search

Customer Table:

The screenshot shows a SQL database interface with the following details:

SQL Editor:

```
30 • create table Customer(email varchar(30) primary key,Cid int,C_name varchar(20),CPhone varchar(10),Address varchar(30),
31             CDOB date,basket_ID int,foreign key(basket_ID) references ShoppingBasket(basket_ID));
32 • insert into Customer values('samruddhikangude@gmail.com',100,'samruddhi','7218380003','Pune', '2017-06-06',298);
33 • insert into Customer values('aditimagar@gmail.com',200,'aditi','8787878787','osmanabad','2015-05-04',898);
34 • insert into Customer values('sohampawar@gmail.com',300,'soham','7765432234','Baramati','2019-09-12',760);
35 • select *from Customer;
36
37 -- Relational Operator --
38 • select *from Books where ISBN <400;
39 • select *from Author where Aid >300;
```

Result Grid:

email	Cid	C_name	CPhone	Address	CDOB	basket_ID
aditimagar@gmail.com	200	aditi	8787878787	osmanabad	2015-05-04	898
samruddhikangude@gmail.com	100	samruddhi	7218380003	Pune	2017-06-06	298
sohampawar@gmail.com	300	soham	7765432234	Baramati	2019-09-12	760
*	HULL	HULL	HULL	HULL	HULL	HULL

Action History:

#	Time	Action	Message
17	14:34:54	select *from ShoppingBasket LIMIT 0, 50000	3 row(s) returned
18	14:35:31	create table Customer(email varchar(30) primary key,Cid int,C_name varchar(20),CPhone varchar(10),Address v...)	0 row(s) affected
19	14:35:31	insert into Customer values('samruddhikangude@gmail.com',100,'samruddhi','7218380003','Pune', '2017-06-06',298)	1 row(s) affected
20	14:35:31	insert into Customer values('aditimagar@gmail.com',200,'aditi','8787878787','osmanabad','2015-05-04',898)	1 row(s) affected
21	14:35:31	insert into Customer values('sohampawar@gmail.com',300,'soham','7765432234','Baramati','2019-09-12',760)	1 row(s) affected
22	14:35:31	select *from Customer LIMIT 0, 50000	3 row(s) returned

Taskbar:

The taskbar includes icons for File Explorer, Mail, OneDrive, Photos, OneNote, Word, Excel, Powerpoint, Publisher, Access, Paint 3D, and Edge.

A: Implementation of relational operators in SQL:

-- Relational Operator --

```
select *from Books where ISBN <400;
select *from Author where Aid >300;
select *from Publisher where Bid =1001;
select *from Books where ISBN <=324;
select *from Author where Aid >=300;
select *from ShoppingBasket where basket_ID !=298;
```

select *from Books where ISBN <400;

The screenshot shows a database management interface with multiple tabs at the top: 'base ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (highlighted in blue), and 'SQL File 10*'. Below the tabs is a toolbar with various icons. The main area displays a series of numbered SQL statements (36 to 44) related to relational operators. Statement 38 is highlighted in blue. The results section shows a table with columns: Bid, year, ISBN, Price, and Title. One row is visible: Bid 3003, year 2021, ISBN 324, Price 1000, Title 'Software Agile'. At the bottom, the 'Books 227' window shows the execution history with actions like creating the 'Author' table and inserting data, along with their timestamps and message counts.

Bid	year	ISBN	Price	Title
3003	2021	324	1000	Software Agile
*	HULL	HULL	HULL	HULL

Books 227 x

Output:

#	Time	Action	Message
26	14:46:42	create table Author(Aname varchar(20) primary key,Aid int,Address varchar(20),Uf varchar(30),Asalary double,...)	0 row(s) affected
27	14:46:42	insert into Author values('Albert',101,'Bengalore','Rickey.com',267654,1001)	1 row(s) affected
28	14:46:42	insert into Author values('Stephen King',202,'Kanada','Stephenking.com',198767,2002)	1 row(s) affected
29	14:46:42	insert into Author values('DrPhillMcCraw',303,'USA','drphillmccraw.com',998642,3003)	1 row(s) affected
30	14:46:42	select * from Author LIMIT 0, 50000	3 row(s) returned
31	14:47:55	select *from Books where ISBN <400 LIMIT 0, 50000	1 row(s) returned

select *from Author where Aid >300;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes tabs for 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (which is selected), and 'SQL File 10*'. Below the menu is a toolbar with various icons. The main area displays a code editor with the following SQL script:

```
36
37  -- Relational Operator --
38 • select *from Books where ISBN <400;
39 • select *from Author where Aid >300;
40 • select *from Publisher where Bid =1001;
41 • select *from Books where ISBN <=324;
42 • select *from Author where Aid >=300;
43 • select *from ShoppingBasket where basket_ID !=298;
44
```

Below the code editor is a 'Result Grid' table with columns: Aname, Aid, Address, Url, Asalary, and Bid. The data row is:

Aname	Aid	Address	Url	Asalary	Bid
DrPhillMcCraw	303	USA	drphillmccraw.com	998642	3003
*	NULL	NULL	NULL	NULL	NULL

At the bottom of the window, there is an 'Output' section with an 'Action Output' table showing log entries for the executed queries.

select *from Publisher where Bid =1001;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes tabs for 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (selected), and 'SQL File 10*'. Below the menu is a toolbar with various icons. The main area displays a code editor with the following SQL script:

```
36
37  -- Relational Operator --
38 • select *from Books where ISBN <400;
39 • select *from Author where Aid >300;
40 • select *from Publisher where Bid =1001;
41 • select *from Books where ISBN <=324;
42 • select *from Author where Aid >=300;
43 • select *from ShoppingBasket where basket_ID !=298;
44
```

Below the code editor is a 'Result Grid' table with columns: pname, Address, PPhone, Url, and Bid. The data row is:

pname	Address	PPhone	Url	Bid
John	Pune	9090987544	john.com	1001
*	NULL	NULL	NULL	NULL

At the bottom of the window, there is an 'Output' section with an 'Action Output' table showing log entries for the executed queries.

select *from Books where ISBN <=324;

```
base ...) SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* × SQL File 10* 
36
37 -- Relational Operator --
38 • select *from Books where ISBN <400;
39 • select *from Author where Aid >300;
40 • select *from Publisher where Bid =1001;
41 • select *from Books where ISBN <=324;
42 • select *from Author where Aid >=300;
43 • select *from ShoppingBasket where basket_ID !=298;
44
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

Bid	year	ISBN	Price	Title
3003	2021	324	1000	Software Agile
*	HULL	HULL	HULL	HULL

Books 230 × | Apply | Revert |

Output:

Action Output

#	Time	Action	Message
29	14:46:42	insert into Author values('DrPhillMcCraw',303,'USA','drphillmccraw.com',998642,3003)	1 row(s) affected
30	14:46:42	select *from Author LIMIT 0, 50000	3 row(s) returned
31	14:47:55	select *from Books where ISBN <400 LIMIT 0, 50000	1 row(s) returned
32	14:48:26	select *from Author where Aid >300 LIMIT 0, 50000	1 row(s) returned
33	14:50:17	select *from Publisher where Bid =1001 LIMIT 0, 50000	1 row(s) returned
34	14:50:47	select *from Books where ISBN <=324 LIMIT 0, 50000	1 row(s) returned

select *from Author where Aid >=300;

```
base ...) SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* × SQL File 10* 
36
37 -- Relational Operator --
38 • select *from Books where ISBN <400;
39 • select *from Author where Aid >300;
40 • select *from Publisher where Bid =1001;
41 • select *from Books where ISBN <=324;
42 • select *from Author where Aid >=300;
43 • select *from ShoppingBasket where basket_ID !=298;
44
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

Aname	Aid	Address	Url	Asalary	Bid
DrPhillMcCraw	303	USA	drphillmccraw.com	998642	3003
*	HULL	HULL	HULL	HULL	HULL

Author 231 × | Apply | Revert |

Output:

Action Output

#	Time	Action	Message
30	14:46:42	select *from Author LIMIT 0, 50000	3 row(s) returned
31	14:47:55	select *from Books where ISBN <400 LIMIT 0, 50000	1 row(s) returned
32	14:48:26	select *from Author where Aid >300 LIMIT 0, 50000	1 row(s) returned
33	14:50:17	select *from Publisher where Bid =1001 LIMIT 0, 50000	1 row(s) returned
34	14:50:47	select *from Books where ISBN <=324 LIMIT 0, 50000	1 row(s) returned
35	14:51:17	select *from Author where Aid >=300 LIMIT 0, 50000	1 row(s) returned

select *from ShoppingBasket where basket_ID !=298;

The screenshot shows a Windows desktop environment with a browser window open to a SQL query interface. The browser title bar includes tabs for "base (...)" and several other databases like "Bank_Management_System", "Food_Factory", etc. The main content area displays a SQL script and its execution results.

SQL Script:

```
36
37 -- Relational Operator --
38 • select *from Books where ISBN <400;
39 • select *from Author where Aid >300;
40 • select *from Publisher where Bid =1001;
41 • select *from Books where ISBN <=324;
42 • select *from Author where Aid >=300;
43 • select *from ShoppingBasket where basket_ID !=298;
44
```

Result Grid:

basket_ID	Bid
898	2002
760	3003
*	NULL

Action Output:

#	Time	Action	Message
31	14:47:55	select *from Books where ISBN <400 LIMIT 0, 50000	1 row(s) returned
32	14:48:26	select *from Author where Aid >300 LIMIT 0, 50000	1 row(s) returned
33	14:50:17	select *from Publisher where Bid =1001 LIMIT 0, 50000	1 row(s) returned
34	14:50:47	select *from Books where ISBN <=324 LIMIT 0, 50000	1 row(s) returned
35	14:51:17	select *from Author where Aid >=300 LIMIT 0, 50000	1 row(s) returned
36	14:51:44	select *from ShoppingBasket where basket_ID !=298 LIMIT 0, 50000	2 row(s) returned

The taskbar at the bottom of the screen shows various pinned icons, including File Explorer, Edge, and several system icons.

B: Boolean operators and pattern matching:

-- Boolean operators --

```
select C_name from Customer where email ='samruddhikangude@gmail.com';  
select basket_ID from ShoppingBasket where Bid=4004;
```

```
select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;  
select C_name from Customer where email='nikita@gmail.com' AND Cid=300;
```

```
select pname from Publisher where Address='Pune' OR Url='abc.com';  
select pname from Publisher where Address='kanada' OR Url='xyz.com';
```

```
select pname from Publisher where Address='Pune' != Url='john.com';
```

```
select C_name from Customer where email ='samruddhikangude@gmail.com';
```

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes 'File', 'Edit', 'View', 'Tools', 'Help', and tabs for 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (which contains the current script), and 'SQL File 10*'. Below the menu is a toolbar with icons for file operations like Open, Save, Print, and a search bar. The main window displays a script editor with numbered lines 44 through 55. Lines 46-47 show basic Boolean operators. Lines 49-50 demonstrate AND conditions. Lines 52-53 demonstrate OR conditions. Line 55 demonstrates a NOT equal condition. To the right of the script is a 'Result Grid' showing a single row for 'C_name' with the value 'samruddhi'. Below the grid is an 'Output' pane titled 'Customer 233' which shows the query 'select C_name from Customer where email ='samruddhikangude@gmail.com' LIMIT 0, 50000' and a message indicating '1 row(s) returned'. The bottom of the screen shows the Windows taskbar with various pinned application icons.

```
44  
45  -- Boolean operators --  
46 •  select C_name from Customer where email ='samruddhikangude@gmail.com';  
47 •  select basket_ID from ShoppingBasket where Bid=4004;  
48  
49 •  select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;  
50 •  select C_name from Customer where email='nikita@gmail.com' AND Cid=300;  
51  
52 •  select pname from Publisher where Address='Pune' OR Url='abc.com';  
53 •  select pname from Publisher where Address='kanada' OR Url='xyz.com';  
54  
55 •  select pname from Publisher where Address='Pune' != Url='john.com';
```

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

C_name
samruddhi

Customer 233 x Read Only

Action Output

#	Time	Action
1	14:59:13	select C_name from Customer where email ='samruddhikangude@gmail.com' LIMIT 0, 50000

Message
1 row(s) returned

select basket_ID from ShoppingBasket where Bid=4004;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* × SQL File 10*  
44  
45 -- Boolean operators --  
46 • select C_name from Customer where email = 'samruddhikangude@gmail.com';  
47 • select basket_ID from ShoppingBasket where Bid=4004;  
48  
49 • select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;  
50 • select C_name from Customer where email='nikita@gmail.com' AND Cid=300;  
51  
52 • select pname from Publisher where Address='Pune' OR Url='abc.com';  
53 • select pname from Publisher where Address='kanada' OR Url='xyz.com';  
54  
55 • select pname from Publisher where Address='Pune' != Url='john.com';
```

The result grid shows one row with the value 'NULL' under the column 'basket_ID'. The output pane shows two actions in the 'Action Output' log:

#	Time	Action	Message
1	14:59:13	select C_name from Customer where email = 'samruddhikangude@gmail.com'	1 row(s) returned
2	15:00:45	select basket_ID from ShoppingBasket	0 row(s) returned

select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* × SQL File 10*  
44  
45 -- Boolean operators --  
46 • select C_name from Customer where email = 'samruddhikangude@gmail.com';  
47 • select basket_ID from ShoppingBasket where Bid=4004;  
48  
49 • select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;  
50 • select C_name from Customer where email='nikita@gmail.com' AND Cid=300;  
51  
52 • select pname from Publisher where Address='Pune' OR Url='abc.com';  
53 • select pname from Publisher where Address='kanada' OR Url='xyz.com';  
54  
55 • select pname from Publisher where Address='Pune' != Url='john.com';
```

The result grid shows one row with the value 'aditi' under the column 'C_name'. The output pane shows three actions in the 'Action Output' log:

#	Time	Action	Message
1	14:59:13	select C_name from Customer where email = 'samruddhikangude@gmail.com' LIMIT 0, 50000	1 row(s) returned
2	15:00:45	select basket_ID from ShoppingBasket where Bid=4004 LIMIT 0, 50000	0 row(s) returned
3	15:01:22	select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200 LIMIT 0, 50000	1 row(s) returned

select C_name from Customer where email='nikita@gmail.com' AND Cid=300;

The screenshot shows a database management interface with multiple tabs at the top: 'Bank_Management_System', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (which is active), and 'SQL File 10*'. Below the tabs is a toolbar with various icons. A code editor window displays the following SQL query:

```
44
45 -- Boolean operators --
46 • select C_name from Customer where email ='samruddhikangude@gmail.com';
47 • select basket_ID from ShoppingBasket where Bid=4004;
48
49 • select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;
50 • select C_name from Customer where email='nikita@gmail.com' AND Cid=300;
51
52 • select pname from Publisher where Address='Pune' OR Url='abc.com';
53 • select pname from Publisher where Address='kanada' OR Url='xyz.com';
54
55 • select pname from Publisher where Address='Pune' != Url='john.com';
```

The result grid shows a single column 'C_name' with one row containing the value 'nikita'. Below the result grid is an 'Action Output' section showing the history of actions taken:

#	Time	Action	Message
1	14:59:13	select C_name from Customer where email ='samruddhikangude@gmail.com' LIMIT 0, 50000	1 row(s) returned
2	15:00:45	select basket_ID from ShoppingBasket where Bid=4004 LIMIT 0, 50000	0 row(s) returned
3	15:01:22	select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200 LIMIT 0, 50000	1 row(s) returned
4	15:01:52	select C_name from Customer where email='nikita@gmail.com' AND Cid=300 LIMIT 0, 50000	0 row(s) returned

select pname from Publisher where Address='Pune' OR Url='abc.com';

The screenshot shows a database management interface with the same tab structure and toolbar as the previous screenshot. The code editor window displays the following SQL query:

```
44
45 -- Boolean operators --
46 • select C_name from Customer where email ='samruddhikangude@gmail.com';
47 • select basket_ID from ShoppingBasket where Bid=4004;
48
49 • select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;
50 • select C_name from Customer where email='nikita@gmail.com' AND Cid=300;
51
52 • select pname from Publisher where Address='Pune' OR Url='abc.com';
53 • select pname from Publisher where Address='kanada' OR Url='xyz.com';
54
55 • select pname from Publisher where Address='Pune' != Url='john.com';
```

The result grid shows a single column 'pname' with one row containing the value 'John'. Below the result grid is an 'Action Output' section showing the history of actions taken:

#	Time	Action	Message
1	14:59:13	select C_name from Customer where email ='samruddhikangude@gmail.com' LIMIT 0, 50000	1 row(s) returned
2	15:00:45	select basket_ID from ShoppingBasket where Bid=4004 LIMIT 0, 50000	0 row(s) returned
3	15:01:22	select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200 LIMIT 0, 50000	1 row(s) returned
4	15:01:52	select C_name from Customer where email='nikita@gmail.com' AND Cid=300 LIMIT 0, 50000	0 row(s) returned
5	15:02:20	select pname from Publisher where Address='Pune' OR Url='abc.com' LIMIT 0, 50000	1 row(s) returned

select pname from Publisher where Address='kanada' OR Url='xyz.com';

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* × SQL File 10*  
44  
45 -- Boolean operators --  
46 • select C_name from Customer where email ='samruddhikangude@gmail.com';  
47 • select basket_ID from ShoppingBasket where Bid=4004;  
48  
49 • select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;  
50 • select C_name from Customer where email='nikita@gmail.com' AND Cid=300;  
51  
52 • select pname from Publisher where Address='Pune' OR Url='abc.com';  
53 • select pname from Publisher where Address='kanada' OR Url='xyz.com';  
54  
55 • select pname from Publisher where Address='Pune' != Url='john.com';
```

The result grid shows one row with the value 'NULL' under the 'pname' column.

Below the query window, the 'Publisher 238' output window displays the execution log:

#	Time	Action	Message
1	14:59:13	select C_name from Customer where email ='samruddhikangude@gmail.com' LIMIT 0, 50000	1 row(s) returned
2	15:00:45	select basket_ID from ShoppingBasket where Bid=4004 LIMIT 0, 50000	0 row(s) returned
3	15:01:22	select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200 LIMIT 0, 50000	1 row(s) returned
4	15:01:52	select C_name from Customer where email='nikita@gmail.com' AND Cid=300 LIMIT 0, 50000	0 row(s) returned
5	15:02:20	select pname from Publisher where Address='Pune' OR Url='abc.com' LIMIT 0, 50000	1 row(s) returned
6	15:02:50	select pname from Publisher where Address='kanada' OR Url='xyz.com' LIMIT 0, 50000	0 row(s) returned

The taskbar at the bottom of the screen includes icons for File Explorer, Search, Task View, File, Settings, Start, Task Manager, Edge, File Explorer, Mail, Dell, Task View, Paint 3D, Snipping Tool, and Google Chrome.

select pname from Publisher where Address='Pune' != Url='john.com';

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* × SQL File 10*  
44  
45 -- Boolean operators --  
46 • select C_name from Customer where email ='samruddhikangude@gmail.com';  
47 • select basket_ID from ShoppingBasket where Bid=4004;  
48  
49 • select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200;  
50 • select C_name from Customer where email='nikita@gmail.com' AND Cid=300;  
51  
52 • select pname from Publisher where Address='Pune' OR Url='abc.com';  
53 • select pname from Publisher where Address='kanada' OR Url='xyz.com';  
54  
55 • select pname from Publisher where Address='Pune' != Url='john.com';
```

The result grid shows two rows with values 'Albert' and 'hidkey' under the 'pname' column.

Below the query window, the 'Publisher 239' output window displays the execution log:

#	Time	Action	Message
2	15:00:45	select basket_ID from ShoppingBasket where Bid=4004 LIMIT 0, 50000	0 row(s) returned
3	15:01:22	select C_name from Customer where email='aditimagar@gmail.com' AND Cid=200 LIMIT 0, 50000	1 row(s) returned
4	15:01:52	select C_name from Customer where email='nikita@gmail.com' AND Cid=300 LIMIT 0, 50000	0 row(s) returned
5	15:02:20	select pname from Publisher where Address='Pune' OR Url='abc.com' LIMIT 0, 50000	1 row(s) returned
6	15:02:50	select pname from Publisher where Address='kanada' OR Url='xyz.com' LIMIT 0, 50000	0 row(s) returned
7	15:03:25	select pname from Publisher where Address='Pune' != Url='john.com' LIMIT 0, 50000	2 row(s) returned

The taskbar at the bottom of the screen includes icons for File Explorer, Search, Task View, File, Settings, Start, Task Manager, Edge, File Explorer, Mail, Dell, Task View, Paint 3D, Snipping Tool, and Google Chrome.

-- Pattern Matching --

```
select *from Publisher where pname LIKE 'h%';
select *from Customer where C_name LIKE '%i';
select *from Customer where C_name LIKE '%d%';
select *from Publisher where pname LIKE '_ b%';
select *from Publisher where pname LIKE '%h_';
select *from Books where Title REGEXP 'a';
```

select *from Publisher where pname LIKE 'h%';

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes tabs for SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9*, and SQL File 10*. Below the menu is a toolbar with various icons. The main pane displays the following SQL code:

```
56
57  -- Pattern Matching --
58 •  select *from Publisher where pname LIKE 'h%';
59 •  select *from Customer where C_name LIKE '%i';
60 •  select *from Customer where C_name LIKE '%d%';
61 •  select *from Publisher where pname LIKE '_ b%';
62 •  select *from Publisher where pname LIKE '%h_';
63 •  select *from Books where Title REGEXP 'a';
64
```

Below the code is a Result Grid showing the output of the query:

pname	Address	PPhone	Url	Bid
hickey	kolkata	7689543221	hickey.com	3003
NULL	NULL	NULL	NULL	NULL

At the bottom, the Output window shows the execution log:

#	Time	Action	Message
1	15:12:11	select *from Publisher where pname LIKE 'h%' LIMIT 0, 50000	1 row(s) returned

select *from Customer where C_name LIKE '%i';

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes tabs for SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9*, and SQL File 10*. Below the menu is a toolbar with various icons. The main pane displays the following SQL code:

```
56
57  -- Pattern Matching --
58 •  select *from Publisher where pname LIKE 'h%';
59 •  select *from Customer where C_name LIKE '%i';
60 •  select *from Customer where C_name LIKE '%d%';
61 •  select *from Publisher where pname LIKE '_ b%';
62 •  select *from Publisher where pname LIKE '%h_';
63 •  select *from Books where Title REGEXP 'a';
64
```

Below the code is a Result Grid showing the output of the query:

email	Cid	C_name	CPhone	Address	CDOB	basket_ID
aditimagar@gmail.com	200	aditi	8787878787	osmanabad	2015-05-04	898
samruddhikangude@gmail.com	100	samruddhi	7218380003	Pune	2017-06-06	298
NULL	NULL	NULL	NULL	NULL	NULL	NULL

At the bottom, the Output window shows the execution log:

#	Time	Action	Message
1	15:12:11	select *from Publisher where pname LIKE 'h%' LIMIT 0, 50000	1 row(s) returned
2	15:13:11	select *from Customer where C_name LIKE '%i' LIMIT 0, 50000	2 row(s) returned

select *from Customer where C_name LIKE '%d%';

The screenshot shows a database management system interface with multiple tabs at the top: SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. Below the tabs is a toolbar with icons for file operations like Open, Save, and Print, along with a limit set to 50,000 rows. The main area contains the following SQL code:

```
56
57 -- Pattern Matching --
58 • select *from Publisher where pname LIKE 'h%';
59 • select *from Customer where C_name LIKE '%i';
60 • select *from Customer where C_name LIKE '%d%';
61 • select *from Publisher where pname LIKE '_b%';
62 • select *from Publisher where pname LIKE '%h_';
63 • select *from Books where Title REGEXP 'a';
64
```

Below the code is a "Result Grid" table with the following data:

email	Cid	C_name	CPhone	Address	CDOB	basket_ID
aditmagar@gmail.com	200	aditi	8787878787	osmanabad	2015-05-04	898
samruddhikangude@gmail.com	100	samruddhi	7218380003	Pune	2017-06-06	298
*	NULL	NULL	NULL	NULL	NULL	NULL

At the bottom of the interface, there is an "Output" section titled "Action Output" showing the history of actions:

#	Time	Action	Message
1	15:12:11	select *from Publisher where pname LIKE 'h%' LIMIT 0, 50000	1 row(s) returned
2	15:13:11	select *from Customer where C_name LIKE '%i' LIMIT 0, 50000	2 row(s) returned
3	15:13:39	select *from Customer where C_name LIKE '%d%' LIMIT 0, 50000	2 row(s) returned

The taskbar at the bottom of the screen is visible, showing various application icons.

select *from Publisher where pname LIKE '__b%';

The screenshot shows a database management system interface with multiple tabs at the top: SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. Below the tabs is a toolbar with icons for file operations like Open, Save, and Print, along with a limit set to 50,000 rows. The main area contains the following SQL code:

```
56
57 -- Pattern Matching --
58 • select *from Publisher where pname LIKE 'h%';
59 • select *from Customer where C_name LIKE '%i';
60 • select *from Customer where C_name LIKE '%d%';
61 • select *from Publisher where pname LIKE '__b%';
62 • select *from Publisher where pname LIKE '%h_';
63 • select *from Books where Title REGEXP 'a';
64
```

Below the code is a "Result Grid" table with the following data:

pname	Address	PPhone	Url	Bid
Albert	mumbai	9865784321	albert.cor	albert.com
*	NULL	NULL	NULL	NULL

At the bottom of the interface, there is an "Output" section titled "Action Output" showing the history of actions:

#	Time	Action	Message
1	15:12:11	select *from Publisher where pname LIKE 'h%' LIMIT 0, 50000	1 row(s) returned
2	15:13:11	select *from Customer where C_name LIKE '%i' LIMIT 0, 50000	2 row(s) returned
3	15:13:39	select *from Customer where C_name LIKE '%d%' LIMIT 0, 50000	2 row(s) returned
4	15:14:06	select *from Publisher where pname LIKE '__b%' LIMIT 0, 50000	1 row(s) returned

The taskbar at the bottom of the screen is visible, showing various application icons.

```
select *from Publisher where pname LIKE '%h__';
```

The screenshot shows a SQL Server Management Studio window. The query editor contains the following code:

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*  
56  
57 -- Pattern Matching --  
58 • select *from Publisher where pname LIKE 'h%';  
59 • select *from Customer where C_name LIKE '%i';  
60 • select *from Customer where C_name LIKE '%d%';  
61 • select *from Publisher where pname LIKE '__b%';  
62 • select *from Publisher where pname LIKE '%h__';  
63 • select *from Books where Title REGEXP 'a';  
64
```

The result grid shows one row of data from the Publisher table:

pname	Address	PPhone	Url	Bid
hichey	kolkata	7689543221	hichey.com	3003

The output pane shows the execution log:

#	Time	Action	Message
1	15:12:11	select *from Publisher where pname LIKE 'h%' LIMIT 0, 50000	1 row(s) returned
2	15:13:11	select *from Customer where C_name LIKE "%i" LIMIT 0, 50000	2 row(s) returned
3	15:13:39	select *from Customer where C_name LIKE "%d%" LIMIT 0, 50000	2 row(s) returned
4	15:14:06	select *from Publisher where pname LIKE '__b%' LIMIT 0, 50000	1 row(s) returned
5	15:14:36	select *from Publisher where pname LIKE "%h__" LIMIT 0, 50000	1 row(s) returned

```
select *from Books where Title REGEXP 'a';
```

The screenshot shows a SQL Server Management Studio window. The query editor contains the following code:

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*  
56  
57 -- Pattern Matching --  
58 • select *from Publisher where pname LIKE 'h%';  
59 • select *from Customer where C_name LIKE '%i';  
60 • select *from Customer where C_name LIKE '%d%';  
61 • select *from Publisher where pname LIKE '__b%';  
62 • select *from Publisher where pname LIKE '%h__';  
63 • select *from Books where Title REGEXP 'a';  
64
```

The result grid shows two rows of data from the Books table:

Bid	year	ISBN	Price	Title
1001	2002	765	2200	Advanced Java
3003	2021	324	1000	Software Agile

The output pane shows the execution log:

#	Time	Action	Message
1	15:12:11	select *from Publisher where pname LIKE 'h%' LIMIT 0, 50000	1 row(s) returned
2	15:13:11	select *from Customer where C_name LIKE "%i" LIMIT 0, 50000	2 row(s) returned
3	15:13:39	select *from Customer where C_name LIKE "%d%" LIMIT 0, 50000	2 row(s) returned
4	15:14:06	select *from Publisher where pname LIKE '__b%' LIMIT 0, 50000	1 row(s) returned
5	15:14:36	select *from Publisher where pname LIKE "%h__" LIMIT 0, 50000	1 row(s) returned
6	15:15:06	select *from Books where Title REGEXP 'a' LIMIT 0, 50000	2 row(s) returned

C: Arithmetic operations and built in functions:

-- Arithmetic operations --

```
select *from Author where (Asalary+10000)<250000;  
select *from Books where (Price-800)<1000;  
select *from Author where (Asalary*10)>300000;  
select *from Books where (Price%10)<150;  
select *from Books where (Price/10)<150;  
select *from Author where (Asalary/2*10)<1000000;
```

select *from Author where (Asalary+10000)<250000;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes 'File', 'Edit', 'View', 'Tools', 'Help', and tabs for 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*', and 'SQL File 10*'. Below the menu is a toolbar with icons for file operations, search, and export. A status bar at the bottom indicates 'Limit to 50000 rows'.

The code editor window contains the following SQL statements:

```
64  
65    -- Arithmetic operations --  
66 •  select *from Author where (Asalary+10000)<250000;  
67 •  select *from Books where (Price-800)<1000;  
68 •  select *from Author where (Asalary*10)>300000;  
69 •  select *from Books where (Price%10)<150;  
70 •  select *from Books where (Price/10)<150;  
71 •  select *from Author where (Asalary/2*10)<1000000;  
72
```

The 'Result Grid' pane displays the output of the first query:

Aname	Aid	Address	Url	Asalary	Bid
Stephen King	202	Kanada	Stephenking.com	198767	2002
NULL	NULL	NULL	NULL	NULL	NULL

The 'Output' pane shows the execution log:

#	Time	Action	Message
2	15:13:11	select *from Customer where C_name LIKE '%a' LIMIT 0, 50000	2 row(s) returned
3	15:13:39	select *from Customer where C_name LIKE '%d%' LIMIT 0, 50000	2 row(s) returned
4	15:14:06	select *from Publisher where pname LIKE '_b%' LIMIT 0, 50000	1 row(s) returned
5	15:14:36	select *from Publisher where pname LIKE '%h__' LIMIT 0, 50000	1 row(s) returned
6	15:15:06	select *from Books where Title REGEXP 'a' LIMIT 0, 50000	2 row(s) returned
7	15:28:46	select *from Author where (Asalary+10000)<250000 LIMIT 0, 50000	1 row(s) returned

select *from Books where (Price-800)<1000;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes 'File', 'Edit', 'View', 'Tools', 'Help', and tabs for 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*', and 'SQL File 10*'. Below the menu is a toolbar with icons for file operations, search, and export. A status bar at the bottom indicates 'Limit to 50000 rows'.

The code editor window contains the following SQL statements:

```
64  
65    -- Arithmetic operations --  
66 •  select *from Author where (Asalary+10000)<250000;  
67 •  select *from Books where (Price-800)<1000;  
68 •  select *from Author where (Asalary*10)>300000;  
69 •  select *from Books where (Price%10)<150;  
70 •  select *from Books where (Price/10)<150;  
71 •  select *from Author where (Asalary/2*10)<1000000;  
72
```

The 'Result Grid' pane displays the output of the second query:

Bid	year	ISBN	Price	Title
2002	2011	987	1770	PostgreSQL
3003	2021	324	1000	Software Agile
NULL	NULL	NULL	NULL	NULL

The 'Output' pane shows the execution log:

#	Time	Action	Message
3	15:13:39	select *from Customer where C_name LIKE '%d%' LIMIT 0, 50000	2 row(s) returned
4	15:14:06	select *from Publisher where pname LIKE '_b%' LIMIT 0, 50000	1 row(s) returned
5	15:14:36	select *from Publisher where pname LIKE '%h__' LIMIT 0, 50000	1 row(s) returned
6	15:15:06	select *from Books where Title REGEXP 'a' LIMIT 0, 50000	2 row(s) returned
7	15:28:46	select *from Author where (Asalary+10000)<250000 LIMIT 0, 50000	1 row(s) returned
8	15:29:12	select *from Books where (Price-800)<1000 LIMIT 0, 50000	2 row(s) returned

```
select *from Author where (Asalary*10)>300000;
```

The screenshot shows a SQL database interface with multiple tabs at the top: SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. Below the tabs is a toolbar with various icons. The main area displays the following SQL code:

```
64
65      -- Arithmetic operations --
66 •  select *from Author where (Asalary+10000)<250000;
67 •  select *from Books where (Price-800)<1000;
68 •  select *from Author where (Asalary*10)>300000;
69 •  select *from Books where (Price%10)<150;
70 •  select *from Books where (Price/10)<150;
71 •  select *from Author where (Asalary/2*10)<1000000;
72
```

Below the code is a "Result Grid" table with columns: Aname, Aid, Address, Url, Asalary, Bid. The data is as follows:

Aname	Aid	Address	Url	Asalary	Bid
Albert	101	Bengalore	Rhickey.com	267654	1001
DrPhillMcCraw	303	USA	drphillmccraw.com	998642	3003
Stephen King	202	Kanada	Stephenking.com	198767	2002
*	NULL	NULL	NULL	NULL	NULL

At the bottom of the interface, there is an "Output" section titled "Author 248" which shows the history of actions taken:

#	Time	Action	Message
4	15:14:06	select *from Publisher where pname LIKE '_b%' LIMIT 0, 50000	1 row(s) returned
5	15:14:36	select *from Publisher where pname LIKE '%h__' LIMIT 0, 50000	1 row(s) returned
6	15:15:06	select *from Books where Title REGEXP 'a' LIMIT 0, 50000	2 row(s) returned
7	15:28:46	select *from Author where (Asalary+10000)<250000 LIMIT 0, 50000	1 row(s) returned
8	15:29:12	select *from Books where (Price-800)<1000 LIMIT 0, 50000	2 row(s) returned
9	15:29:39	select *from Author where (Asalary*10)>300000 LIMIT 0, 50000	3 row(s) returned

```
select *from Books where (Price%10)<150;
```

The screenshot shows a SQL database interface with multiple tabs at the top: SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. Below the tabs is a toolbar with various icons. The main area displays the following SQL code:

```
64
65      -- Arithmetic operations --
66 •  select *from Author where (Asalary+10000)<250000;
67 •  select *from Books where (Price-800)<1000;
68 •  select *from Author where (Asalary*10)>300000;
69 •  select *from Books where (Price%10)<150;
70 •  select *from Books where (Price/10)<150;
71 •  select *from Author where (Asalary/2*10)<1000000;
72
```

Below the code is a "Result Grid" table with columns: Bid, year, ISBN, Price, Title. The data is as follows:

Bid	year	ISBN	Price	Title
1001	2002	765	2200	Advanced Java
2002	2011	987	1770	Postgree SQL
3003	2021	324	1000	Software Agile
*	NULL	NULL	NULL	NULL

At the bottom of the interface, there is an "Output" section titled "Books 249" which shows the history of actions taken:

#	Time	Action	Message
5	15:14:36	select *from Publisher where pname LIKE '%h__' LIMIT 0, 50000	1 row(s) returned
6	15:15:06	select *from Books where Title REGEXP 'a' LIMIT 0, 50000	2 row(s) returned
7	15:28:46	select *from Author where (Asalary+10000)<250000 LIMIT 0, 50000	1 row(s) returned
8	15:29:12	select *from Books where (Price-800)<1000 LIMIT 0, 50000	2 row(s) returned
9	15:29:39	select *from Author where (Asalary*10)>300000 LIMIT 0, 50000	3 row(s) returned
10	15:30:08	select *from Books where (Price%10)<150 LIMIT 0, 50000	3 row(s) returned

select *from Books where (Price/10)<150;

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*  
64  
65      -- Arithmetic operations --  
66 •  select *from Author where (Asalary+10000)<250000;  
67 •  select *from Books where (Price-800)<1000;  
68 •  select *from Author where (Asalary*10)>300000;  
69 •  select *from Books where (Price%10)<150;  
70 •  select *from Books where (Price/10)<150;  
71 •  select *from Author where (Asalary/2*10)<1000000;  
72
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

Bid	year	ISBN	Price	Title
3003	2021	324	1000	Software Agile

Action Output

#	Time	Action	Message
7	15:28:46	select *from Author where (Asalary+10000)<250000 LIMIT 0, 50000	1 row(s) returned
8	15:29:12	select *from Books where (Price-800)<1000 LIMIT 0, 50000	2 row(s) returned
9	15:29:39	select *from Author where (Asalary*10)>300000 LIMIT 0, 50000	3 row(s) returned
10	15:30:08	select *from Books where (Price%10)<150 LIMIT 0, 50000	3 row(s) returned
11	15:30:33	select *from Books where (Price/10)<150 LIMIT 0, 50000	1 row(s) returned
12	15:30:43	select *from Books where (Price/10)<150 LIMIT 0, 50000	1 row(s) returned

select *from Author where (Asalary/2*10)<1000000;

```
base )... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*  
64  
65      -- Arithmetic operations --  
66 •  select *from Author where (Asalary+10000)<250000;  
67 •  select *from Books where (Price-800)<1000;  
68 •  select *from Author where (Asalary*10)>300000;  
69 •  select *from Books where (Price%10)<150;  
70 •  select *from Books where (Price/10)<150;  
71 •  select *from Author where (Asalary/2*10)<1000000;  
72
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

Aname	Aid	Address	Url	Asalary	Bid
Stephen King	202	Kanada	Stephenking.com	198767	2002

Action Output

#	Time	Action	Message
8	15:29:12	select *from Books where (Price-800)<1000 LIMIT 0, 50000	2 row(s) returned
9	15:29:39	select *from Author where (Asalary*10)>300000 LIMIT 0, 50000	3 row(s) returned
10	15:30:08	select *from Books where (Price%10)<150 LIMIT 0, 50000	3 row(s) returned
11	15:30:33	select *from Books where (Price/10)<150 LIMIT 0, 50000	1 row(s) returned
12	15:30:43	select *from Books where (Price/10)<150 LIMIT 0, 50000	1 row(s) returned
13	15:31:17	select *from Author where (Asalary/2*10)<1000000 LIMIT 0, 50000	1 row(s) returned

-- built in functions --

```
select lower(pname) from Publisher;
select upper(pname) from Publisher;
select sum(Price) from Books;
select max(Asalary) from Author;
select min(Asalary) from Author;
select sqrt(9);
select floor(9.4);
select ceil(3.6);
select power(9,2);
select strcmp('samruddhi','aditi');
select char_length('samruddhi');
select concat('Samruddhi',' kangude');
select instr('AIML DEPARTMENT','L');
select mid('AIML DEPARTMENT IN ALARD',5,11);
select reverse('SAMRUDDHI');
select space(100);
```

select lower(pname) from Publisher;

The screenshot shows the Oracle SQL Developer interface with several tabs at the top: base(...), SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. Below the tabs is a toolbar with various icons for file operations like Open, Save, Print, and Database navigation.

The code editor area displays the following SQL statements:

```
73  -- built in functions --
74 •  select lower(pname) from Publisher;
75 •  select upper(pname) from Publisher;
76 •  select sum(Price) from Books;
77 •  select max(Asalary) from Author;
78 •  select min(Asalary) from Author;
79 •  select sqrt(9);
80 •  select floor(9.4);
81 •  select ceil(3.6);
82 •  select power(9,2);
```

The Result Grid shows the output for the first query:

lower(pname)
john
albert
hickey

The Output pane shows the log entry for the query:

#	Time	Action	Message
1	15:45:50	select lower(pname) from Publisher LIMIT 0, 50000	3 row(s) returned

The system tray at the bottom of the window includes icons for the Windows Start button, Search, Task View, File Explorer, Google Drive, Mail, Edge, Firefox, Word, Excel, OneNote, Photos, and Google Chrome.

select upper(pname) from Publisher;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
73    -- built in functions --
74 •  select lower(pname) from Publisher;
75 •  select upper(pname) from Publisher;
76 •  select sum(Price) from Books;
77 •  select max(Asalary) from Author;
78 •  select min(Asalary) from Author;
79 •  select sqrt(9);
80 •  select floor(9.4);
81 •  select ceil(3.6);
82 •  select power(9,2);
```

The result grid shows the output for the selected query:

upper(pname)
JOHN
ALBERT
HICHEY

The status bar at the bottom indicates "Result 254" and "Read Only".

select sum(Price) from Books;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
73    -- built in functions --
74 •  select lower(pname) from Publisher;
75 •  select upper(pname) from Publisher;
76 •  select sum(Price) from Books;
77 •  select max(Asalary) from Author;
78 •  select min(Asalary) from Author;
79 •  select sqrt(9);
80 •  select floor(9.4);
81 •  select ceil(3.6);
82 •  select power(9,2);
```

The result grid shows the output for the selected query:

sum(Price)
4970

The status bar at the bottom indicates "Result 255" and "Read Only".

select max(Asalary) from Author;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
73    -- built in functions --
74 •  select lower(pname) from Publisher;
75 •  select upper(pname) from Publisher;
76 •  select sum(Price) from Books;
77 •  select max(Asalary) from Author; -- This line is highlighted
78 •  select min(Asalary) from Author;
79 •  select sqrt(9);
80 •  select floor(9.4);
81 •  select ceil(3.6);
82 •  select power(9,2);
```

The result grid shows the output of the highlighted query:

max(Asalary)
998642

The status bar at the bottom indicates "Result 256" and "Read Only".

select min(Asalary) from Author;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
73    -- built in functions --
74 •  select lower(pname) from Publisher;
75 •  select upper(pname) from Publisher;
76 •  select sum(Price) from Books;
77 •  select max(Asalary) from Author;
78 •  select min(Asalary) from Author; -- This line is highlighted
79 •  select sqrt(9);
80 •  select floor(9.4);
81 •  select ceil(3.6);
82 •  select power(9,2);
```

The result grid shows the output of the highlighted query:

min(Asalary)
198767

The status bar at the bottom indicates "Result 257" and "Read Only".

select sqrt(9);

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
73  -- built in functions --
74 •  select lower(pname) from Publisher;
75 •  select upper(pname) from Publisher;
76 •  select sum(Price) from Books;
77 •  select max(Asalary) from Author;
78 •  select min(Asalary) from Author;
79 •  select sqrt(9);
80 •  select floor(9.4);
81 •  select ceil(3.6);
82 •  select power(9,2);
```

The result grid shows the output of the selected query:

sqrt(9)
3

The status bar at the bottom indicates "Result 258" and "Read Only".

select floor(9.4);

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
73  -- built in functions --
74 •  select lower(pname) from Publisher;
75 •  select upper(pname) from Publisher;
76 •  select sum(Price) from Books;
77 •  select max(Asalary) from Author;
78 •  select min(Asalary) from Author;
79 •  select sqrt(9);
80 •  select floor(9.4);
81 •  select ceil(3.6);
82 •  select power(9,2);
```

The result grid shows the output of the selected query:

floor(9.4)
9

The status bar at the bottom indicates "Result 259" and "Read Only".

select ceil(3.6);

The screenshot shows the SQL Server Management Studio interface. The query window contains the following SQL code:

```
73    -- built in functions --
74 •    select lower(pname) from Publisher;
75 •    select upper(pname) from Publisher;
76 •    select sum(Price) from Books;
77 •    select max(Asalary) from Author;
78 •    select min(Asalary) from Author;
79 •    select sqrt(9);
80 •    select floor(9.4);
81 •    select ceil(3.6);
82 •    select power(9,2);
```

The line `81 • select ceil(3.6);` is highlighted. The results grid shows the output of the query:

Result Grid
ceil(3.6)
4

The results pane displays the output of the query:

Output

Action Output

#	Time	Action	Message
3	15:46:57	select sum(Price) from Books LIMIT 0, 50000	1 row(s) returned
4	15:51:37	select max(Asalary) from Author LIMIT 0, 50000	1 row(s) returned
5	15:52:04	select min(Asalary) from Author LIMIT 0, 50000	1 row(s) returned
6	15:52:27	select sqrt(9) LIMIT 0, 50000	1 row(s) returned
7	15:53:00	select floor(9.4) LIMIT 0, 50000	1 row(s) returned
8	15:53:38	select ceil(3.6) LIMIT 0, 50000	1 row(s) returned

Read Only

select power(9,2);

The screenshot shows the SQL Server Management Studio interface. The query window contains the following SQL code:

```
74 •    select lower(pname) from Publisher;
75 •    select upper(pname) from Publisher;
76 •    select sum(Price) from Books;
77 •    select max(Asalary) from Author;
78 •    select min(Asalary) from Author;
79 •    select sqrt(9);
80 •    select floor(9.4);
81 •    select ceil(3.6);
82 •    select power(9,2);
83 •    select strcmp('samruddhi','aditi');
```

The line `82 • select power(9,2);` is highlighted. The results grid shows the output of the query:

Result Grid
power(9,2)
81

The results pane displays the output of the query:

Output

Action Output

#	Time	Action	Message
4	15:51:37	select max(Asalary) from Author LIMIT 0, 50000	1 row(s) returned
5	15:52:04	select min(Asalary) from Author LIMIT 0, 50000	1 row(s) returned
6	15:52:27	select sqrt(9) LIMIT 0, 50000	1 row(s) returned
7	15:53:00	select floor(9.4) LIMIT 0, 50000	1 row(s) returned
8	15:53:38	select ceil(3.6) LIMIT 0, 50000	1 row(s) returned
9	15:54:04	select power(9,2) LIMIT 0, 50000	1 row(s) returned

Read Only

```
select strcmp('samruddhi','aditi');
```

The screenshot shows a SQL query editor interface with multiple tabs at the top. The active tab is 'SQL File 9*'. The query window contains the following code:

```
81 •    select ceil(3.6);
82 •    select power(9,2);
83 •    select strcmp('samruddhi','aditi');
84 •    select char_length('samruddhi');
85 •    select concat('Samruddhi', ' kangude');
86 •    select instr('AIML DEPARTMENT','L');
87 •    select mid('AIML DEPARTMENT IN ALARD',5,11);
88 •    select reverse('SAMRUDDHI');
89 •    select space(100);
90
```

The result grid shows the output of the selected query:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
strcmp('samruddhi','aditi')			
1			

Below the result grid is a 'Result 262' pane with an 'Output' section. It displays the following log:

Action Output	#	Time	Action	Message
	5	15:52:04	select min(Asalary) from Author LIMIT 0, 50000	1 row(s) returned
	6	15:52:27	select sqrt(9) LIMIT 0, 50000	1 row(s) returned
	7	15:53:00	select floor(9.4) LIMIT 0, 50000	1 row(s) returned
	8	15:53:38	select ceil(3.6) LIMIT 0, 50000	1 row(s) returned
	9	15:54:04	select power(9,2) LIMIT 0, 50000	1 row(s) returned
	10	15:54:42	select strcmp('samruddhi','aditi') LIMIT 0, 50000	1 row(s) returned

The system tray icons are visible at the bottom of the screen.

```
select char_length('samruddhi');
```

The screenshot shows a SQL query editor interface with multiple tabs at the top. The active tab is 'SQL File 9*'. The query window contains the following code:

```
81 •    select ceil(3.6);
82 •    select power(9,2);
83 •    select strcmp('samruddhi','aditi');
84 •    select char_length('samruddhi');
85 •    select concat('Samruddhi', ' kangude');
86 •    select instr('AIML DEPARTMENT','L');
87 •    select mid('AIML DEPARTMENT IN ALARD',5,11);
88 •    select reverse('SAMRUDDHI');
89 •    select space(100);
90
```

The result grid shows the output of the selected query:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
char_length('samruddhi')			
9			

Below the result grid is a 'Result 263' pane with an 'Output' section. It displays the following log:

Action Output	#	Time	Action	Message
	6	15:52:27	select sqrt(9) LIMIT 0, 50000	1 row(s) returned
	7	15:53:00	select floor(9.4) LIMIT 0, 50000	1 row(s) returned
	8	15:53:38	select ceil(3.6) LIMIT 0, 50000	1 row(s) returned
	9	15:54:04	select power(9,2) LIMIT 0, 50000	1 row(s) returned
	10	15:54:42	select strcmp('samruddhi','aditi') LIMIT 0, 50000	1 row(s) returned
	11	15:55:09	select char_length('samruddhi') LIMIT 0, 50000	1 row(s) returned

The system tray icons are visible at the bottom of the screen.

```
select concat('Samruddhi',' kangude');
```

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled "SQL File 5*", "Bank_Management_System", "SQL File 5*", "Food_Factory", "SQL File 7", "SQL File 8*", "SQL File 9*", and "SQL File 10*". The "SQL File 9*" tab is active. The code area contains the following SQL statements:

```
81 •    select ceil(3.6);
82 •    select power(9,2);
83 •    select strcmp('samruddhi','aditi');
84 •    select char_length('samruddhi');
85 •    select concat('Samruddhi',' kangude');
86 •    select instr('AIML DEPARTMENT','L');
87 •    select mid('AIML DEPARTMENT IN ALARD',5,11);
88 •    select reverse('SAMRUDDHI');
89 •    select space(100);
90
```

The result grid shows the output of the last statement:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
concat('Samruddhi',' kangude')			
▶ Samruddhi kangude			

Below the result grid is a "Result 264" section with an "Output" tab. It displays the following log of actions:

#	Time	Action	Message
7	15:53:00	select floor(9.4) LIMIT 0, 50000	1 row(s) returned
8	15:53:38	select ceil(3.6) LIMIT 0, 50000	1 row(s) returned
9	15:54:04	select power(9,2) LIMIT 0, 50000	1 row(s) returned
10	15:54:42	select strcmp('samruddhi','aditi') LIMIT 0, 50000	1 row(s) returned
11	15:55:09	select char_length('samruddhi') LIMIT 0, 50000	1 row(s) returned
12	15:55:43	select concat('Samruddhi','kangude') LIMIT 0, 50000	1 row(s) returned

The system tray icons are visible at the bottom of the screen.

```
select instr('AIML DEPARTMENT','L');
```

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled "SQL File 5*", "Bank_Management_System", "SQL File 5*", "Food_Factory", "SQL File 7", "SQL File 8*", "SQL File 9*", and "SQL File 10*". The "SQL File 9*" tab is active. The code area contains the following SQL statements:

```
81 •    select ceil(3.6);
82 •    select power(9,2);
83 •    select strcmp('samruddhi','aditi');
84 •    select char_length('samruddhi');
85 •    select concat('Samruddhi',' kangude');
86 •    select instr('AIML DEPARTMENT','L');
87 •    select mid('AIML DEPARTMENT IN ALARD',5,11);
88 •    select reverse('SAMRUDDHI');
89 •    select space(100);
90
```

The result grid shows the output of the last statement:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
instr('AIML DEPARTMENT','L')			
▶ 4			

Below the result grid is a "Result 265" section with an "Output" tab. It displays the following log of actions:

#	Time	Action	Message
8	15:53:38	select ceil(3.6) LIMIT 0, 50000	1 row(s) returned
9	15:54:04	select power(9,2) LIMIT 0, 50000	1 row(s) returned
10	15:54:42	select strcmp('samruddhi','aditi') LIMIT 0, 50000	1 row(s) returned
11	15:55:09	select char_length('samruddhi') LIMIT 0, 50000	1 row(s) returned
12	15:55:43	select concat('Samruddhi','kangude') LIMIT 0, 50000	1 row(s) returned
13	15:56:32	select instr('AIML DEPARTMENT','L') LIMIT 0, 50000	1 row(s) returned

The system tray icons are visible at the bottom of the screen.

```
select mid('AIML DEPARTMENT IN ALARD',5,11);
```

The screenshot shows a SQL query editor interface. At the top, there are tabs for various databases: base ... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*. Below the tabs, the SQL code is displayed:

```
81 • select ceil(3.6);
82 • select power(9,2);
83 • select strcmp('samruddhi','aditi');
84 • select char_length('samruddhi');
85 • select concat('Samruddhi',' kangude');
86 • select instr('AIML DEPARTMENT','L');
87 • select mid('AIML DEPARTMENT IN ALARD',5,11);
88 • select reverse('SAMRUDDHI');
89 • select space(100);
90
```

The line `87 • select mid('AIML DEPARTMENT IN ALARD',5,11);` is highlighted. In the result grid below, the output is shown in a single row:

mid('AIML DEPARTMENT IN ALARD',5,11)

Below the result grid, the output pane shows the execution log:

Result 266 x Read Only

Output

Action Output

#	Time	Action	Message
9	15:54:04	select power(9.2) LIMIT 0, 50000	1 row(s) returned
10	15:54:42	select strcmp('samruddhi','aditi') LIMIT 0, 50000	1 row(s) returned
11	15:55:09	select char_length('samruddhi') LIMIT 0, 50000	1 row(s) returned
12	15:55:43	select concat('Samruddhi',' kangude') LIMIT 0, 50000	1 row(s) returned
13	15:56:32	select instr('AIML DEPARTMENT','L') LIMIT 0, 50000	1 row(s) returned
14	16:00:43	select mid('AIML DEPARTMENT IN ALARD',5,11) LIMIT 0, 50000	1 row(s) returned

At the bottom, the taskbar shows various application icons.

```
select reverse('SAMRUDDHI');
```

The screenshot shows a SQL query editor interface. At the top, there are tabs for various databases: base ... SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*. Below the tabs, the SQL code is displayed:

```
81 • select ceil(3.6);
82 • select power(9,2);
83 • select strcmp('samruddhi','aditi');
84 • select char_length('samruddhi');
85 • select concat('Samruddhi',' kangude');
86 • select instr('AIML DEPARTMENT','L');
87 • select mid('AIML DEPARTMENT IN ALARD',5,11);
88 • select reverse('SAMRUDDHI');
89 • select space(100);
90
```

The line `88 • select reverse('SAMRUDDHI');` is highlighted. In the result grid below, the output is shown in a single row:

reverse('SAMRUDDHI')

Below the result grid, the output pane shows the execution log:

Result 267 x Read Only

Output

Action Output

#	Time	Action	Message
10	15:54:42	select strcmp('samruddhi','aditi') LIMIT 0, 50000	1 row(s) returned
11	15:55:09	select char_length('samruddhi') LIMIT 0, 50000	1 row(s) returned
12	15:55:43	select concat('Samruddhi',' kangude') LIMIT 0, 50000	1 row(s) returned
13	15:56:32	select instr('AIML DEPARTMENT','L') LIMIT 0, 50000	1 row(s) returned
14	16:00:43	select mid('AIML DEPARTMENT IN ALARD',5,11) LIMIT 0, 50000	1 row(s) returned
15	16:01:10	select reverse('SAMRUDDHI') LIMIT 0, 50000	1 row(s) returned

At the bottom, the taskbar shows various application icons.

select space(100);

The screenshot shows a SQL query editor interface with multiple tabs at the top: SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (selected), and SQL File 10*. The main area displays the following SQL code:

```
81 •    select ceil(3.6);
82 •    select power(9,2);
83 •    select strcmp('samruddhi','aditi');
84 •    select char_length('samruddhi');
85 •    select concat('Samruddhi',' kngude');
86 •    select instr('AIML DEPARTMENT','L');
87 •    select mid('AIML DEPARTMENT IN ALARD',5,11);
88 •    select reverse('SAMRUDDHI');
89 •    select space(100);
90
```

Below the code, a Result Grid shows the output of the last query:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
space(100)	...		

The Output pane below shows the execution log:

Action	Time	Action	Message
11	15:55:09	select char_length(samruddhi) LIMIT 0, 50000	1 row(s) returned
12	15:55:43	select concat('Samruddhi','kngude') LIMIT 0, 50000	1 row(s) returned
13	15:56:32	select instr('AIML DEPARTMENT','L') LIMIT 0, 50000	1 row(s) returned
14	16:00:43	select mid('AIML DEPARTMENT IN ALARD',5,11) LIMIT 0, 50000	1 row(s) returned
15	16:01:10	select reverse('SAMRUDDHI') LIMIT 0, 50000	1 row(s) returned
16	16:01:36	select space(100) LIMIT 0, 50000	1 row(s) returned

The system tray icons are visible at the bottom of the window.

D: Group Functions:

-- Group functions--

select Aname from Author group by Aname;
select Aname,count(*)from Author group by Aname;
select Aname from Author where Asalary<=270000 group by Aname;
select Title,sum(Price),avg(Price) from Books group by Title;
select Title,min(Price),max(Price)from Books group by Title;

select Aname from Author group by Aname;

The screenshot shows a SQL database interface with multiple tabs at the top: base (...), SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. Below the tabs is a toolbar with various icons. The main area contains the following SQL code:

```
91
92 -- Group functions--
93 • select Aname from Author group by Aname;
94 • select Aname, count(*) from Author group by Aname;
95 • select Aname from Author where Asalary<=270000 group by Aname;
96 • select Title, sum(Price), avg(Price) from Books group by Title;
97 • select Title, min(Price), max(Price) from Books group by Title;
98
99
100
```

Below the code is a Result Grid table with one column labeled "Aname". The data rows are:

Aname
Albert
Stephen King
DrPhillMcCraw
* NULL

At the bottom, there is an "Output" section titled "Author 269" showing a log of actions:

#	Time	Action	Message
1	16:37:16	select Aname from Author group by Aname LIMIT 0, 50000	3 row(s) returned

The interface includes a taskbar at the bottom with various application icons.

select Aname,count(*)from Author group by Aname;

The screenshot shows the SQL Server Management Studio interface. The top menu bar has tabs for 'base ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (which is selected), and 'SQL File 10*'. Below the menu is a toolbar with various icons. The main pane displays the following SQL code:

```
91
92    -- Group functions--
93 • select Aname from Author group by Aname;
94 • select Aname, count(*) from Author group by Aname;
95 • select Aname from Author where Asalary<=270000 group by Aname;
96 • select Title,sum(Price),avg(Price) from Books group by Title;
97 • select Title,min(Price),max(Price)from Books group by Title;
98
99
100
```

Below the code is a 'Result Grid' table with two columns: 'Aname' and 'count(*)'. The data shows three rows: Albert (1), DrPhillMcCraw (1), and Stephen King (1).

The status bar at the bottom indicates 'Result 270' and 'Read Only'.

select Aname from Author where Asalary<=270000 group by Aname;

The screenshot shows the SQL Server Management Studio interface. The top menu bar has tabs for 'base ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (selected), and 'SQL File 10*'. Below the menu is a toolbar with various icons. The main pane displays the same SQL code as the previous screenshot, but the highlighted line is the one filtering by salary.

```
91
92    -- Group functions--
93 • select Aname from Author group by Aname;
94 • select Aname, count(*) from Author group by Aname;
95 • select Aname from Author where Asalary<=270000 group by Aname;
96 • select Title,sum(Price),avg(Price) from Books group by Title;
97 • select Title,min(Price),max(Price)from Books group by Title;
98
99
100
```

Below the code is a 'Result Grid' table with one column 'Aname'. It shows two rows: Albert and Stephen King. There is also a row labeled 'NULL'.

The status bar at the bottom indicates 'Author 271'.

select Title,sum(Price),avg(Price) from Books group by Title;

The screenshot shows a SQL database interface with multiple tabs at the top: SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. The main area contains the following SQL code:

```
91
92      -- Group functions--
93 •  select Aname from Author group by Aname;
94 •  select Aname,count(*)from Author group by Aname;
95 •  select Aname from Author where Asalary<=270000 group by Aname;
96 •  select Title,sum(Price),avg(Price) from Books group by Title; [SELECTED]
97 •  select Title,min(Price),max(Price)from Books group by Title;
98
99
100
```

Below the code is a Result Grid table:

Title	sum(Price)	avg(Price)
Advanced Java	2200	2200
Postgree SQL	1770	1770
Software Agile	1000	1000

The Output section shows the following Action History:

#	Time	Action	Message
1	16:37:16	select Aname from Author group by Aname LIMIT 0, 50000	3 row(s) returned
2	16:38:05	select Aname,count(*)from Author group by Aname LIMIT 0, 50000	3 row(s) returned
3	16:38:34	select Aname from Author where Asalary<=270000 group by Aname LIMIT 0, 50000	2 row(s) returned
4	16:39:56	select Title,sum(Price),avg(Price) from Books group by Title LIMIT 0, 50000	3 row(s) returned

select Title,min(Price),max(Price)from Books group by Title;

The screenshot shows a SQL database interface with multiple tabs at the top: SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 9* (highlighted in blue), and SQL File 10*. The main area contains the following SQL code:

```
91
92      -- Group functions--
93 •  select Aname from Author group by Aname;
94 •  select Aname,count(*)from Author group by Aname;
95 •  select Aname from Author where Asalary<=270000 group by Aname;
96 •  select Title,sum(Price),avg(Price) from Books group by Title;
97 •  select Title,min(Price),max(Price)from Books group by Title; [SELECTED]
98
99
100
```

Below the code is a Result Grid table:

Title	min(Price)	max(Price)
Advanced Java	2200	2200
Postgree SQL	1770	1770
Software Agile	1000	1000

The Output section shows the following Action History:

#	Time	Action	Message
1	16:37:16	select Aname from Author group by Aname LIMIT 0, 50000	3 row(s) returned
2	16:38:05	select Aname,count(*)from Author group by Aname LIMIT 0, 50000	3 row(s) returned
3	16:38:34	select Aname from Author where Asalary<=270000 group by Aname LIMIT 0, 50000	2 row(s) returned
4	16:39:56	select Title,sum(Price),avg(Price) from Books group by Title LIMIT 0, 50000	3 row(s) returned
5	16:40:23	select Title,min(Price),max(Price)from Books group by Title LIMIT 0, 50000	3 row(s) returned

E: Processing Date and Time Functions:

-- Processing Date --

```
SELECT curdate();
SELECT current_timestamp();
SELECT NOW();
SELECT DATE(NOW());
SELECT MONTH('2022-12-31');
SELECT MONTHNAME(NOW());
SELECT DAYNAME("2032-12-12");
SELECT MONTHNAME("2020-05-06");
SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

SELECT curdate();

The screenshot shows a SQL database management system interface. At the top, there are several tabs labeled 'SQL FILE 1' through 'SQL FILE 10'. Below the tabs is a toolbar with various icons. The main area contains a code editor with numbered lines of SQL code. The code includes comments and examples for each function. Below the code editor is a 'Result Grid' table with one row showing the output of the 'curdate()' function. At the bottom, there is an 'Output' section showing the execution log with a single entry for the 'curdate()' query.

Result Grid
curdate()
▶ 2023-04-27

Output

Action Output
Time Action
1 19:50:15 SELECT curdate() LIMIT 0, 50000

Message
1 row(s) returned

```
SELECT current_timestamp();
```

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
base )_ SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* X SQL File 10* SQ
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

The Result Grid shows the output of the current_timestamp() function:

current_timestamp()
2023-04-27 19:50:50

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
Result 301 x
Output
Action Output
# Time Action
1 19:50:15 SELECT curdate() LIMIT 0, 50000
2 19:50:50 SELECT current_timestamp() LIMIT 0, 50000
Message
1 row(s) returned
1 row(s) returned
```

The Result Grid shows the output of the NOW() function:

NOW()
2023-04-27 19:51:20

```
SELECT NOW();
```

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
base )_ SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* X SQL File 10* SQ
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

The Result Grid shows the output of the NOW() function:

NOW()
2023-04-27 19:51:20

The Result 302 window shows the execution log:

#	Time	Action	Message
1	19:50:15	SELECT curdate() LIMIT 0, 50000	1 row(s) returned
2	19:50:50	SELECT current_timestamp() LIMIT 0, 50000	1 row(s) returned
3	19:51:20	SELECT NOW() LIMIT 0, 50000	1 row(s) returned

SELECT DATE(NOW());

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

The result grid shows the output of the query:

DATE(NOW())
2023-04-27

The output pane shows the execution log:

#	Time	Action	Message
1	19:50:15	SELECT curdate() LIMIT 0, 50000	1 row(s) returned
2	19:50:50	SELECT current_timestamp() LIMIT 0, 50000	1 row(s) returned
3	19:51:20	SELECT NOW() LIMIT 0, 50000	1 row(s) returned
4	19:51:51	SELECT DATE(NOW()) LIMIT 0, 50000	1 row(s) returned

SELECT MONTH('2022-12-31');

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

The result grid shows the output of the query:

MONTH('2022-12-31')
12

The output pane shows the execution log:

#	Time	Action	Message
1	19:50:15	SELECT curdate() LIMIT 0, 50000	1 row(s) returned
2	19:50:50	SELECT current_timestamp() LIMIT 0, 50000	1 row(s) returned
3	19:51:20	SELECT NOW() LIMIT 0, 50000	1 row(s) returned
4	19:51:51	SELECT DATE(NOW()) LIMIT 0, 50000	1 row(s) returned
5	19:52:17	SELECT MONTH('2022-12-31') LIMIT 0, 50000	1 row(s) returned

```
SELECT MONTHNAME(NOW());
```

The screenshot shows a SQL query editor interface. At the top, there are tabs for "base ...", "SQL File 5*", "Bank_Management_System", "SQL File 5*", "Food_Factory", "SQL File 7", "SQL File 8*", "SQL File 9*", and "SQL File 10*". The "SQL File 9*" tab is active. Below the tabs, a code editor displays the following SQL code:

```
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

Below the code editor is a "Result Grid" section with the following output:

MONTHNAME(NOW())
April

The screenshot shows the same SQL query editor interface. The "SQL File 9*" tab is still active. Below the tabs, a code editor displays the following SQL code:

```
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

Below the code editor is a "Result Grid" section with the following output:

DAYNAME("2032-12-12")
Sunday

The screenshot shows the same SQL query editor interface. The "SQL File 9*" tab is still active. Below the tabs, a code editor displays the following SQL code:

```
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

Below the code editor is a "Result Grid" section with the following output:

MONTHNAME(NOW())
April

The screenshot shows the same SQL query editor interface. The "SQL File 9*" tab is still active. Below the tabs, a code editor displays the following SQL code:

```
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

Below the code editor is a "Result Grid" section with the following output:

DAYNAME("2032-12-12")
Sunday

```
SELECT MONTHNAME("2020-05-06");
```

The screenshot shows a SQL database interface with multiple tabs at the top: 'base ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*', and 'SQL File 10*'. The current tab is 'SQL File 9*'. The query window contains the following code:

```
98 -- Processing Date --
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

The result grid shows one row of data:

MONTHNAME("2020-05-06")	May
-------------------------	-----

The status bar at the bottom indicates 'Result 307' and 'Read Only'.

```
SELECT CID ,DATE(CDOB) FROM CUSTOMER;
```

The screenshot shows a SQL database interface with multiple tabs at the top: 'base ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*', and 'SQL File 10*'. The current tab is 'SQL File 9*'. The query window contains the following code:

```
99 • SELECT curdate();
100 • SELECT current_timestamp();
101 • SELECT NOW();
102 • SELECT DATE(NOW());
103 • SELECT MONTH('2022-12-31');
104 • SELECT MONTHNAME(NOW());
105 • SELECT DAYNAME("2032-12-12");
106 • SELECT MONTHNAME("2020-05-06");
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;
108
```

The result grid shows three rows of data:

CID	DATE(CDOB)
200	2015-05-04
100	2017-06-06
300	2019-09-12

The status bar at the bottom indicates 'Result 308' and 'Read Only'.

-- Time functions --

```
SELECT CURTIME();
SELECT TIME(NOW());
SELECT TIME("19:30:10");
SELECT TIME("2017-08-15 19:30:10");
select current_timestamp();
select time_to_sec(20000);
```

SELECT CURTIME();

The screenshot shows the SQL Server Management Studio interface. The query window displays the following code:

```
108
109  -- Time functions --
110 •  SELECT CURTIME();
111 •  SELECT TIME(NOW());
112 •  SELECT TIME("19:30:10");
113 •  SELECT TIME("2017-08-15 19:30:10");
114 •  select current_timestamp();
115 •  select time_to_sec(20000);
116
117
```

The result grid shows the output of the CURTIME() function:

CURTIME()
19:56:07

The results pane shows the following output:

Action Output
Time Action
1 19:56:07 SELECT CURTIME() LIMIT 0, 50000

Message: 1 row(s) returned

SELECT TIME(NOW());

The screenshot shows the SQL Server Management Studio interface. The query window displays the following code:

```
108
109  -- Time functions --
110 •  SELECT CURTIME();
111 •  SELECT TIME(NOW());
112 •  SELECT TIME("19:30:10");
113 •  SELECT TIME("2017-08-15 19:30:10");
114 •  select current_timestamp();
115 •  select time_to_sec(20000);
116
117
```

The result grid shows the output of the TIME(NOW()) function:

TIME(NOW())
19:56:33

The results pane shows the following output:

Action Output
Time Action
1 19:56:07 SELECT CURTIME() LIMIT 0, 50000
2 19:56:33 SELECT TIME(NOW()) LIMIT 0, 50000

Message: 1 row(s) returned
Message: 1 row(s) returned

```
SELECT TIME("19:30:10");
```

The screenshot shows a Windows desktop environment. At the top is a taskbar with icons for File Explorer, Task View, Start, and several pinned applications. Below the taskbar is a window titled "Result Grid" showing the output of the SQL query. The query itself is in a code editor at the top of the window, and the result grid shows a single row with the value "2023-04-27 19:51:20". Below the result grid is a "Result 302" section showing the execution log with three entries, all of which returned 1 row(s) returned.

```
base )_ SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*  
98 -- Processing Date --  
99 • SELECT curdate();  
100 • SELECT current_timestamp();  
101 • SELECT NOW();  
102 • SELECT DATE(NOW());  
103 • SELECT MONTH('2022-12-31');  
104 • SELECT MONTHNAME(NOW());  
105 • SELECT DAYNAME("2032-12-12");  
106 • SELECT MONTHNAME("2020-05-06");  
107 • SELECT CID ,DATE(CDOB) FROM CUSTOMER;  
  
Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:   
NOW()  
▶ 2023-04-27 19:51:20  
  
Result 302 x Read Only  
Output  
Action Output  
# Time Action Message  
1 19:50:15 SELECT curdate() LIMIT 0, 50000 1 row(s) returned  
2 19:50:50 SELECT current_timestamp() LIMIT 0, 50000 1 row(s) returned  
3 19:51:20 SELECT NOW() LIMIT 0, 50000 1 row(s) returned  
  
Windows taskbar:
```

```
SELECT TIME("2017-08-15 19:30:10");
```

The screenshot shows a Windows desktop environment. At the top is a taskbar with icons for File Explorer, Task View, Start, and several pinned applications. Below the taskbar is a window titled "Result Grid" showing the output of the SQL query. The query itself is in a code editor at the top of the window, and the result grid shows a single row with the value "19:30:10". Below the result grid is a "Result 312" section showing the execution log with four entries, all of which returned 1 row(s) returned.

```
base )_ SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10*  
108  
109 -- Time functions --  
110 • SELECT CURTIME();  
111 • SELECT TIME(NOW());  
112 • SELECT TIME("19:30:10");  
113 • SELECT TIME("2017-08-15 19:30:10");  
114 • select current_timestamp();  
115 • select time_to_sec(20000);  
116  
117  
  
Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:   
TIME("2017-08-15  
19:30:10")  
▶ 19:30:10  
  
Result 312 x Read Only  
Output  
Action Output  
# Time Action Message  
1 19:56:07 SELECT CURTIME() LIMIT 0, 50000 1 row(s) returned  
2 19:56:33 SELECT TIME(NOW()) LIMIT 0, 50000 1 row(s) returned  
3 19:56:58 SELECT TIME("19:30:10") LIMIT 0, 50000 1 row(s) returned  
4 19:57:25 SELECT TIME("2017-08-15 19:30:10") LIMIT 0, 50000 1 row(s) returned  
  
Windows taskbar:
```

```
select current_timestamp();
```

The screenshot shows a SQL query editor interface. At the top, there are tabs for various databases and files. The main area contains the following SQL code:

```
108
109  -- Time functions --
110 •  SELECT CURTIME();
111 •  SELECT TIME(NOW());
112 •  SELECT TIME("19:30:10");
113 •  SELECT TIME("2017-08-15 19:30:10");
114 •  select current_timestamp();
115 •  select time_to_sec(20000);
116
117
```

Below the code, the results are displayed in a grid:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
current_timestamp()			
▶ 2023-04-27 19:57:50			

At the bottom, an 'Output' section shows the execution log:

Action Output	#	Time	Action	Message
	1	19:56:07	SELECT CURTIME() LIMIT 0, 50000	1 row(s) returned
	2	19:56:33	SELECT TIME(NOW()) LIMIT 0, 50000	1 row(s) returned
	3	19:56:58	SELECT TIME("19:30:10") LIMIT 0, 50000	1 row(s) returned
	4	19:57:25	SELECT TIME("2017-08-15 19:30:10") LIMIT 0, 50000	1 row(s) returned
	5	19:57:50	select current_timestamp() LIMIT 0, 50000	1 row(s) returned

```
select time_to_sec(20000);
```

The screenshot shows a SQL query editor interface. At the top, there are tabs for various databases and files. The main area contains the following SQL code:

```
108
109  -- Time functions --
110 •  SELECT CURTIME();
111 •  SELECT TIME(NOW());
112 •  SELECT TIME("19:30:10");
113 •  SELECT TIME("2017-08-15 19:30:10");
114 •  select current_timestamp();
115 •  select time_to_sec(20000);
116
117
```

Below the code, the results are displayed in a grid:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
time_to_sec(20000)			
▶ 7200			

At the bottom, an 'Output' section shows the execution log:

Action Output	#	Time	Action	Message
	1	19:56:07	SELECT CURTIME() LIMIT 0, 50000	1 row(s) returned
	2	19:56:33	SELECT TIME(NOW()) LIMIT 0, 50000	1 row(s) returned
	3	19:56:58	SELECT TIME("19:30:10") LIMIT 0, 50000	1 row(s) returned
	4	19:57:25	SELECT TIME("2017-08-15 19:30:10") LIMIT 0, 50000	1 row(s) returned
	5	19:57:50	select current_timestamp() LIMIT 0, 50000	1 row(s) returned
	6	19:58:15	select time_to_sec(20000) LIMIT 0, 50000	1 row(s) returned

F: Complex Queries and Set Operators:

-- Complex queries--

select *from Books;

select price,Bid,year,ISBN,Title from Books where BId=1001;

select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books;

select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN;

select Title,Bid,Price,ISBN,year,count(*) from Books GROUP BY Title,Bid,Price,ISBN,year Having Title="Advanced Java";

select *from Books;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes 'File', 'Edit', 'View', 'Tools', 'Help', and tabs for 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (which is active), and 'SQL File 10*'. Below the menu is a toolbar with various icons. The main query window displays numbered SQL statements (116 to 125) related to complex queries on the 'Books' table. Statement 118 is highlighted. The results grid shows the output of statement 125, which retrieves data for books with titles 'Advanced Java', 'Postgree SQL', and 'Software Agile'. The bottom section shows the 'Action Output' pane with a history of actions and their messages.

Bid	year	ISBN	Price	Title
1001	2002	765	2200	Advanced Java
2002	2011	987	1770	Postgree SQL
3003	2021	324	1000	Software Agile
*	HULL	HULL	HULL	HULL

Books 315 x

Output:

#	Time	Action	Message
2	19:56:33	SELECT TIME(NOW()) LIMIT 0, 50000	1 row(s) returned
3	19:56:58	SELECT TIME("19:30:10") LIMIT 0, 50000	1 row(s) returned
4	19:57:25	SELECT TIME("2017-08-15 19:30:10") LIMIT 0, 50000	1 row(s) returned
5	19:57:50	select current_timestamp() LIMIT 0, 50000	1 row(s) returned
6	19:58:15	select time_to_sec(20000) LIMIT 0, 50000	1 row(s) returned
7	19:59:27	select * from Books LIMIT 0, 50000	3 row(s) returned

select price,Bid,year,ISBN,Title from Books where BId=1001;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
116
117  -- Complex queries--
118 • select *from Books;
119 • select price,Bid,year,ISBN,Title from Books where BId=1001;
120 • select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books;
121 • select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN;
122 • select Title,Bid,Price,ISBN,year,count(*) from Books
123           GROUP BY Title,Bid,Price,ISBN,year Having Title="Advanced Java";
124
125
```

The result grid shows the following data:

price	Bid	year	ISBN	Title
2200	1001	2002	765	Advanced Java
*	NULL	NULL	NULL	NULL

The status bar at the bottom indicates "Books 316".

The Action Output pane shows the following log entries:

#	Time	Action	Message
3	19:56:58	SELECT TIME("19:30:10") LIMIT 0, 50000	1 row(s) returned
4	19:57:25	SELECT TIME("2017-08-15 19:30:10") LIMIT 0, 50000	1 row(s) returned
5	19:57:50	select current_timestamp() LIMIT 0, 50000	1 row(s) returned
6	19:58:15	select time_to_sec(20000) LIMIT 0, 50000	1 row(s) returned
7	19:59:27	select *from Books LIMIT 0, 50000	3 row(s) returned
8	19:59:56	select price,Bid,year,ISBN,Title from Books where BId=1001 LIMIT 0, 50000	1 row(s) returned

select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
116
117  -- Complex queries--
118 • select *from Books;
119 • select price,Bid,year,ISBN,Title from Books where BId=1001;
120 • select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books;
121 • select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN;
122 • select Title,Bid,Price,ISBN,year,count(*) from Books
123           GROUP BY Title,Bid,Price,ISBN,year Having Title="Advanced Java";
124
125
```

The result grid shows the following data:

Bid	year	Price	Price
1001	2002	2200	1
2002	2011	1770	1
3003	2021	1000	1

The status bar at the bottom indicates "Result 317".

The Action Output pane shows the following log entries:

#	Time	Action	Message
4	19:57:25	SELECT TIME("2017-08-15 19:30:10") LIMIT 0, 50000	1 row(s) returned
5	19:57:50	select current_timestamp() LIMIT 0, 50000	1 row(s) returned
6	19:58:15	select time_to_sec(20000) LIMIT 0, 50000	1 row(s) returned
7	19:59:27	select *from Books LIMIT 0, 50000	3 row(s) returned
8	19:59:56	select price,Bid,year,ISBN,Title from Books where BId=1001 LIMIT 0, 50000	1 row(s) returned
9	20:00:27	select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books	3 row(s) returned

select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN;

```
base ...) SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10* 
116
117 -- Complex queries--
118 • select *from Books;
119 • select price,Bid,year,ISBN,Title from Books where BId=1001;
120 • select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books;
121 • select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN;
122 • select Title,Bid,Price,ISBN,year,count(*) from Books
123           GROUP BY Title,Bid,Price,ISBN,year Having Title="Advanced Java";
124
125
```

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

Title	count>Title)	Bid	ISBN
Advanced Java	1	1001	765
Postgree SQL	1	2002	987

Result 318 x Read Only

Output:

Action Output

#	Time	Action	Message
5	19:57:50	select current_timestamp() LIMIT 0, 50000	1 row(s) returned
6	19:58:15	select time_to_sec(20000) LIMIT 0, 50000	1 row(s) returned
7	19:59:27	select *from Books LIMIT 0, 50000	3 row(s) returned
8	19:59:56	select price,Bid,year,ISBN,Title from Books where BId=1001 LIMIT 0, 50000	1 row(s) returned
9	20:00:27	select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books	3 row(s) returned
10	20:00:58	select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN LIMIT 0, 50000	2 row(s) returned

select Title,Bid,Price,ISBN,year,count(*) from Books GROUP BY Title,Bid,Price,ISBN,year Having Title="Advanced Java";

```
base ...) SQL File 5* Bank_Management_System SQL File 5* Food_Factory SQL File 7 SQL File 8* SQL File 9* SQL File 10* 
116
117 -- Complex queries--
118 • select *from Books;
119 • select price,Bid,year,ISBN,Title from Books where BId=1001;
120 • select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books;
121 • select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN;
122 • select Title,Bid,Price,ISBN,year,count(*) from Books
123           GROUP BY Title,Bid,Price,ISBN,year Having Title="Advanced Java";
124
125
```

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

Title	Bid	Price	ISBN	year	count(*)
Advanced Java	1001	2200	765	2002	1

Result 319 x Read Only

Output:

Action Output

#	Time	Action	Message
6	19:58:15	select time_to_sec(20000) LIMIT 0, 50000	1 row(s) returned
7	19:59:27	select *from Books LIMIT 0, 50000	3 row(s) returned
8	19:59:56	select price,Bid,year,ISBN,Title from Books where BId=1001 LIMIT 0, 50000	1 row(s) returned
9	20:00:27	select Bid ,year ,Price,RANK() Over (partition by year ORDER BY Price DESC) as Price from Books	3 row(s) returned
10	20:00:58	select Title,count>Title),Bid,ISBN from Books where price>1500 GROUP BY Title,Bid,ISBN LIMIT 0, 50000	2 row(s) returned
11	20:01:29	select Title,Bid,Price,ISBN,year,count(*) from Books GROUP BY Title,Bid,Price,ISBN,year Having Tit...	1 row(s) returned

-- set operators--

```
select *from Author;
select *from publisher;
select Aname from Author UNION select pname from publisher;
select Aname from Author UNION ALL select pname from publisher; -- It will take all duplicate contents--
```

select *from Author;

The screenshot shows the SSMS interface with a script window containing the following SQL code:

```
124
125  -- set operators--
126 •  select *from Author;
127 •  select *from publisher;
128 •  select Aname from Author UNION select pname from publisher;
129 •  select Aname from Author UNION ALL select pname from publisher; -- It will take all duplicate contents--
130 •  select distinct Aname,Aid from Author
131    inner join publisher
132      on Author.Aid=publisher.pname;
133
```

Below the script window is a Result Grid displaying the query results:

Aname	Aid	Address	Url	Asalary	Bid
Albert	101	Bengalore	Rhickey.com	267654	1001
DrPhilMcCraw	303	USA	drphilmcraw.com	998642	3003
Stephen King	202	Kanada	Stephenking.com	198767	2002
*	HULL	HULL	HULL	HULL	HULL

At the bottom of the screen, the taskbar is visible with various application icons.

select *from publisher;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
124
125      -- set operators--
126 •  select *from Author;
127 •  select *from publisher;
128 •  select Aname from Author UNION select pname from publisher;
129 •  select Aname from Author UNION ALL select pname from publisher; -- It will take all duplicate contents--
130 •  select distinct Aname,Aid from Author
131     inner join publisher
132       on Author.Aid=publisher.pname;
133
```

The result grid shows the following data:

pname	Address	PPhone	Url	Bid
Albert	mumbai	9865784321	albert.com	2002
hichay	kolkata	7689543221	hichay.com	3003
John	Pune	9090987544	john.com	1001
*	NULL	NULL	NULL	NULL

The output pane shows the following log entries:

#	Time	Action	Message
1	20:03:29	select *from Author LIMIT 0, 50000	3 row(s) returned
2	20:04:05	select *from publisher LIMIT 0, 50000	3 row(s) returned

select Aname from Author UNION select pname from publisher;

The screenshot shows the SQL Server Management Studio interface. The query window contains the following code:

```
124
125      -- set operators--
126 •  select *from Author;
127 •  select *from publisher;
128 •  select Aname from Author UNION select pname from publisher;
129 •  select Aname from Author UNION ALL select pname from publisher; -- It will take all duplicate contents--
130 •  select distinct Aname,Aid from Author
131     inner join publisher
132       on Author.Aid=publisher.pname;
133
```

The result grid shows the following data:

Aname
Albert
Stephen King
DrPhillMcCraw
John
hichay

The output pane shows the following log entries:

#	Time	Action	Message
1	20:03:29	select *from Author LIMIT 0, 50000	3 row(s) returned
2	20:04:05	select *from publisher LIMIT 0, 50000	3 row(s) returned
3	20:04:29	select Aname from Author UNION select pname from publisher	5 row(s) returned

select Aname from Author UNION ALL select pname from publisher; -- It will take all duplicate contents--

The screenshot shows a SQL Server Management Studio (SSMS) interface. At the top, there's a toolbar with various icons like Save, Undo, Redo, and Search. Below the toolbar is a tab bar with several open connections: 'base ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 9*' (which is the active tab), and 'SQL File 10*'. The main area contains a query window with the following code:

```
124  
125    -- set operators--  
126 •  select *from Author;  
127 •  select *from publisher;  
128 •  select Aname from Author UNION select pname from publisher;  
129 •  select Aname from Author UNION ALL select pname from publisher; -- It will take all duplicate contents--  
130 •  select distinct Aname,Aid from Author  
131     inner join publisher  
132       on Author.Aid=publisher.pname;  
133
```

Below the code, there's a 'Result Grid' section showing the output of the last query. The grid has one column labeled 'Aname' and contains the following data:

Aname
Albert
Stephen King
DrPhillMcCraw
John
Albert

At the bottom of the result grid, it says 'Result 323 x' and 'Read Only'. Below the result grid is an 'Output' section with a table titled 'Action Output'. The table has columns '#', 'Time', 'Action', and 'Message'. It lists four actions:

#	Time	Action	Message
1	20:03:29	select *from Author LIMIT 0, 50000	3 row(s) returned
2	20:04:05	select *from publisher LIMIT 0, 50000	3 row(s) returned
3	20:04:29	select Aname from Author UNION select pname from publisher	5 row(s) returned
4	20:04:58	select Aname from Author UNION ALL select pname from publisher	6 row(s) returned

The bottom of the screen shows a taskbar with various icons for Windows, search, file explorer, messaging, folder, Google Sheets, email, Dell logo, Microsoft Edge, Firefox, Word, Excel, Paint, and other system icons.

Group B: MySQL

5. Execute DDL/DML statements which demonstrate the use of views. Update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables.

Creating a database and tables:

```
create database Employee;
use Employee;
```

```
create table Emp1(ID int ,NAME varchar(55),EXPERIENCE int);
insert into Emp1 values(1,'Samruddhi',12);
insert into Emp1 values(2,'Aditi',7);
insert into Emp1 values(3,'Soham',9);
insert into Emp1 values(4,'Nikita',11);
select *from Emp1;
```

The screenshot shows the MySQL Workbench interface. At the top, there are tabs for 'Bank_Management_System' and other SQL files. Below the tabs, a code editor displays the SQL statements used to create the database and table, and insert data into them. The 'Result Grid' section shows the resulting data in a table:

ID	NAME	EXPERIENCE
1	Samruddhi	12
2	Aditi	7
3	Soham	9
4	Nikita	11

Below the result grid is the 'Action Output' log, which lists the execution details for each statement:

#	Time	Action	Message
3	00:16:32	create table Emp1(ID int ,NAME varchar(55),EXPERIENCE int)	0 row(s) affected
4	00:16:32	insert into Emp1 values(1,'Samruddhi',12)	1 row(s) affected
5	00:16:32	insert into Emp1 values(2,'Aditi',7)	1 row(s) affected
6	00:16:32	insert into Emp1 values(3,'Soham',9)	1 row(s) affected
7	00:16:32	insert into Emp1 values(4,'Nikita',11)	1 row(s) affected
8	00:16:32	select *from Emp1 LIMIT 0, 50000	4 row(s) returned

```
create table Emp2(ID int,COUNTRY varchar(10));
insert into Emp2 values(1,'India');
insert into Emp2 values(2,'USA');
insert into Emp2 values(3,'Canada');
select *from Emp2;
```

The screenshot shows a SQL interface with the following details:

- SQL Editor:** Displays the SQL code for creating the 'Emp2' table and inserting three rows. The code is numbered from 25 to 34.
- Result Grid:** Shows the data in the 'Emp2' table with three rows: (1, India), (2, USA), and (3, Canada).
- Action Output:** Shows the execution history with the following log entries:

#	Time	Action	Message
8	00:16:32	select *from Emp2 LIMIT 0, 50000	4 row(s) returned
9	00:17:25	create table Emp2(ID int,COUNTRY varchar(10))	0 row(s) affected
10	00:17:25	insert into Emp2 values(1,'India')	1 row(s) affected
11	00:17:25	insert into Emp2 values(2,'USA')	1 row(s) affected
12	00:17:25	insert into Emp2 values(3,'Canada')	1 row(s) affected
13	00:17:25	select *from Emp2 LIMIT 0, 50000	3 row(s) returned

Taskbar: At the bottom, there is a standard Windows-style taskbar with icons for File Explorer, Google Chrome, and other system utilities.

Creating a multiple views:

create view Emp1view

as

select *from Emp1;

select *from Emp1view;

The screenshot shows the SQL Server Management Studio interface. At the top, there are tabs for 'create database ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 10*', 'SQL File 9*', and 'SQL File 11*'. Below the tabs, a toolbar with various icons is visible. A status bar at the bottom indicates 'Limit to 50000 rows'.

In the main pane, the following SQL code is displayed:

```
10  
11 •  create view Emp1view  
12   as  
13     select *from Emp1;  
14 •  select *from Emp1view;  
15  
16 •  create view Emp1viewA  
17   as  
18     select ID,NAME from Emp1;  
19 •  select *from Emp1viewA;  
20
```

Below the code, a 'Result Grid' shows the data from the 'Emp1viewA' view:

ID	NAME	EXPERIENCE
1	Samruddhi	12
2	Aditi	7
3	Soham	9
4	Nikita	11

At the bottom, a separate window titled 'Emp1view 37 x' displays the execution history:

#	Time	Action	Message
10	00:17:25	insert into Emp2 values(1,'India')	1 row(s) affected
11	00:17:25	insert into Emp2 values(2,'USA')	1 row(s) affected
12	00:17:25	insert into Emp2 values(3,'Canada')	1 row(s) affected
13	00:17:25	select *from Emp2 LIMIT 0, 50000	3 row(s) returned
14	00:21:01	create view Emp1view as select *from Emp1	0 row(s) affected
15	00:21:01	select *from Emp1view LIMIT 0, 50000	4 row(s) returned

The taskbar at the bottom of the screen includes icons for Start, Search, File Explorer, Task View, Mail, Google Drive, Dell, Edge, Firefox, Chrome, Word, Excel, and a few others.

```
create view Emp1viewA
as
select ID,NAME from Emp1;
select *from Emp1viewA;
```

The screenshot shows a SQL management interface with the following details:

- Top Bar:** Shows tabs for '(create database ...)', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 10*', 'SQL File 9*', and 'SQL File 11*'.
- Code Editor:** Displays the following SQL code:

```
14 • select *from Emp1view;
15
16 • create view Emp1viewA
17   as
18   select ID,NAME from Emp1;
19 • select *from Emp1viewA;
20
21 • create view Emp1viewB
22   as
23   select ID,NAME,EXPERIENCE from Emp1 where EXPERIENCE>10;
```
- Result Grid:** Shows a table with columns 'ID' and 'NAME'. The data is:

ID	NAME
1	Samruddhi
2	Aditi
3	Soham
4	Nikita
- Action Output:** Shows a log of actions with their times and messages:

#	Time	Action	Message
12	00:17:25	insert into Emp2 values(3,'Canada')	1 row(s) affected
13	00:17:25	select *from Emp2 LIMIT 0, 50000	3 row(s) returned
14	00:21:01	create view Emp1view as select *from Emp1	0 row(s) affected
15	00:21:01	select *from Emp1view LIMIT 0, 50000	4 row(s) returned
16	00:21:35	create view Emp1viewA as select ID,NAME from Emp1	0 row(s) affected
17	00:21:35	select *from Emp1viewA LIMIT 0, 50000	4 row(s) returned
- Taskbar:** Shows various system icons including the Start button, Search, File Explorer, Task View, Mail, Google Drive, Dell logo, Microsoft Edge, Firefox, Google Chrome, File History, Task Manager, and a network icon.

```
create view Emp1viewB
as
select ID,NAME,EXPERIENCE from Emp1 where EXPERIENCE>10;
select *from Emp1viewB;
```

The screenshot shows a SQL database management interface with the following details:

- SQL Editor:** Displays the following SQL code:

```
18  select ID,NAME from Emp1;
19 •  select *from Emp1viewA;
20
21 •  create view Emp1viewB
22    as
23      select ID,NAME,EXPERIENCE from Emp1 where EXPERIENCE>10;
24 •  select *from Emp1viewB;
25
26 •  create table Emp2(ID int,COUNTRY varchar(10));
27 •  insert into Emp2 values(1,'India');
```
- Result Grid:** Shows the data returned by the query in step 24:

ID	NAME	EXPERIENCE
1	Samruddhi	12
4	Nikita	11
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message
14	00:21:01	create view Emp1view as select *from Emp1	0 row(s) affected
15	00:21:01	select *from Emp1view LIMIT 0, 50000	4 row(s) returned
16	00:21:35	create view Emp1viewA as select ID,NAME from Emp1	0 row(s) affected
17	00:21:35	select *from Emp1viewA LIMIT 0, 50000	4 row(s) returned
18	00:22:08	create view Emp1viewB as select ID,NAME,EXPERIENCE from Emp1 where EXPERIENCE>10	0 row(s) affected
19	00:22:08	select *from Emp1viewB LIMIT 0, 50000	2 row(s) returned
- Taskbar:** Shows various system icons including Start, Search, File Explorer, OneDrive, Mail, Task View, and others.

```

create view Emp2viewA
as
select Emp1.ID,NAME,COUNTRY
from Emp1,Emp2
where Emp1.ID=Emp2.ID;
select *from Emp2viewA;

```

The screenshot shows a SQL interface with multiple tabs at the top: (create database)..., SQL File 5*, Bank_Management_System, SQL File 5*, Food_Factory, SQL File 7, SQL File 8*, SQL File 10*, SQL File 9*, and SQL File 11*. The SQL File 11* tab is active.

The code in the editor is:

```

33
34
35 • create view Emp2viewA
36 as
37 select Emp1.ID,NAME,COUNTRY
38 from Emp1,Emp2
39 where Emp1.ID=Emp2.ID;
40 • select *from Emp2viewA;
41
42 -- DDL Commands On View (create,drop,alter,truncate,rename)--
-- . . .

```

The Result Grid shows the data returned by the query:

ID	NAME	COUNTRY
1	Samruddhi	India
2	Aditi	USA
3	Soham	Canada

The Action Output pane shows the history of actions:

#	Time	Action	Message
16	00:21:35	create view Emp1viewA as select ID,NAME from Emp1	0 row(s) affected
17	00:21:35	select *from Emp1viewA LIMIT 0, 50000	4 row(s) returned
18	00:22:08	create view Emp1viewB as select ID,NAME,EXPERIENCE from Emp1 where EXPERIENCE>10	0 row(s) affected
19	00:22:08	select *from Emp1viewB LIMIT 0, 50000	2 row(s) returned
20	00:22:53	create view Emp2viewA as select Emp1.ID,NAME,COUNTRY from Emp1,Emp2 where Emp1.ID=Emp2.ID	0 row(s) affected
21	00:22:53	select *from Emp2viewA LIMIT 0, 50000	3 row(s) returned

The taskbar at the bottom includes icons for Start, Search, File Explorer, Task View, Mail, Google Drive, Edge, Firefox, Chrome, Microsoft Word, Microsoft Excel, Microsoft Powerpoint, Microsoft OneNote, Microsoft Paint, and Microsoft Snipping Tool.

DDL commands on view(create,drop,alter,truncate,rename):

Create command:

```
create view Emp2viewB  
as  
select ID,COUNTRY from Emp2 where ID>2;  
select *from Emp2viewB;
```

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes options like '(create database)...', 'SQL File 5*', 'Bank_Management_System', etc. The main window displays the following code:

```
41  
42 -- DDL Commands On View (create,drop,alter,truncate,rename)--  
43 -- Create Command on view--  
44 • create view Emp2viewB  
45 as  
46 select ID,COUNTRY from Emp2 where ID>2;  
47 • select *from Emp2viewB;  
48  
49 -- drop Command on view--  
50 • drop view Emp1view;  
-- . . . . .
```

Below the code, there is a 'Result Grid' pane showing the output of the 'select *from Emp2viewB' command:

ID	COUNTRY
3	Canada

At the bottom, the 'Output' pane shows the execution log with the following entries:

#	Time	Action	Message
18	00:22:08	create view Emp1viewB as select ID,NAME,EXPERIENCE from Emp1 where EXPERIENCE>10	0 row(s) affected
19	00:22:08	select *from Emp1viewB LIMIT 0, 50000	2 row(s) returned
20	00:22:53	create view Emp2viewA as select Emp1.ID,NAME,COUNTRY from Emp1,Emp2 where Emp1.ID=Emp2.ID	0 row(s) affected
21	00:22:53	select *from Emp2viewA LIMIT 0, 50000	3 row(s) returned
22	00:27:39	create view Emp2viewB as select ID,COUNTRY from Emp2 where ID>2	0 row(s) affected
23	00:27:39	select *from Emp2viewB LIMIT 0, 50000	1 row(s) returned

Drop command:

```
drop view Emp1view;
drop view Emp1viewA;
drop view Emp1viewB;
drop view Emp2viewA;
drop view Emp2viewB;
```

The screenshot shows a SQL Management Studio environment. The top part is a query editor window titled 'SQL File 11*' containing a series of SQL commands. The commands include selecting from 'Emp2' where ID > 2, dropping various views (Emp1view, Emp1viewA, Emp1viewB, Emp2viewA, Emp2viewB), altering a view to select Name, EXPERIENCE from 'Emp1' where EXPERIENCE > 9, truncating the 'Emp1' table, and renaming a view. The bottom part is an 'Output' window titled 'Action Output' which displays the results of the executed commands, showing successful completion for each action.

#	Time	Action	Message
23	00:27:39	select *from Emp2viewB LIMIT 0, 50000	1 row(s) returned
24	00:29:37	drop view Emp1view	0 row(s) affected
25	00:29:37	drop view Emp1viewA	0 row(s) affected
26	00:29:37	drop view Emp1viewB	0 row(s) affected
27	00:29:37	drop view Emp2viewA	0 row(s) affected
28	00:29:37	drop view Emp2viewB	0 row(s) affected

Alter command:

Alter view Emp1view AS select Name,EXPERIENCE from Emp1 where EXPERIENCE>9;
select *from Emp1view;

The screenshot shows a SQL management interface with the following details:

- SQL Editor:** Displays the following SQL code:

```
54 • drop view Emp2viewB;
55
56 -- alter command on view--
57 • Alter view Emp1view AS select Name,EXPERIENCE from Emp1 where EXPERIENCE>9;
58 • select *from Emp1view;
59
60 -- truncate command on view--
61 • truncate table Emp1;
62
63 -- Rename command in view--
```
- Result Grid:** Shows a table with two rows of data:

Name	EXPERIENCE
Samruddhi	12
Nikita	11
- Action Output:** Shows a log of 25 actions with their times and messages:

#	Time	Action	Message
20	00:33:36	create view Emp2viewA as select Emp1.ID,NAME,COUNTRY from Emp1,Emp2 where Emp1.ID=Emp2.ID	0 row(s) affected
21	00:33:36	select *from Emp2viewA LIMIT 0, 50000	3 row(s) returned
22	00:33:47	create view Emp2viewB as select ID,COUNTRY from Emp2 where ID>2	0 row(s) affected
23	00:33:47	select *from Emp2viewB LIMIT 0, 50000	1 row(s) returned
24	00:33:56	Alter view Emp1view AS select Name,EXPERIENCE from Emp1 where EXPERIENCE>9	0 row(s) affected
25	00:33:56	select *from Emp1view LIMIT 0, 50000	2 row(s) returned

Truncate command:

```
truncate table Emp1;
```

The screenshot shows a Windows desktop environment. At the top, there's a taskbar with various pinned icons including File Explorer, Microsoft Edge, and several other application icons. Below the taskbar is a window titled '(create database ...)' which contains a SQL script. The script includes several comments and commands related to views and tables, with the 'truncate table Emp1;' command highlighted in blue. Below the script, there's an 'Output' section showing a table of log entries with columns for '#', 'Time', 'Action', and 'Message'. The log entries show the execution of various database operations, including creating and altering views, selecting from them, and finally truncating the 'Emp1' table. The bottom right corner of the screen shows the Windows Start button.

```
54 • drop view Emp2viewB;
55
56 -- alter command on view--
57 • Alter view Emp1view AS select Name,EXPERIENCE from Emp1 where EXPERIENCE>9;
58 • select *from Emp1view;
59
60 -- truncate command on view--
61 • truncate table Emp1;
62
63 -- Rename command in view--
64 • rename table Emp1 to Emp3;
65 • select *from Emp3;
66 • rename table Emp3 to Emp1;
67 • select *from Emp1;
68
69 -- DML Commands On View (insert,update,delete)--
70 -- Insert command on view--
71 • insert into Emp1viewB values(6,'Prachi',21);
```

#	Time	Action	Message
21	00:33:36	select *from Emp2viewA LIMIT 0, 50000	3 row(s) returned
22	00:33:47	create view Emp2viewB as select ID,COUNTRY from Emp2 where ID>2	0 row(s) affected
23	00:33:47	select *from Emp2viewB LIMIT 0, 50000	1 row(s) returned
24	00:33:56	Alter view Emp1view AS select Name,EXPERIENCE from Emp1 where EXPERIENCE>9	0 row(s) affected
25	00:33:56	select *from Emp1view LIMIT 0, 50000	2 row(s) returned
26	00:34:31	truncate table Emp1	0 row(s) affected

Rename command:

rename table Emp1 to Emp3;

select *from Emp3;

The screenshot shows a SQL query window with the following content:

```
60 -- truncate command on view--
61 • truncate table Emp1;
62
63 -- Rename command in view--
64 • rename table Emp1 to Emp3;
65 • select *from Emp3;
66 • rename table Emp3 to Emp1;
67 • select *from Emp1;
68
69 -- DML Commands On View (insert,update,delete)--
70
```

Below the code, there is a Result Grid showing the following data:

ID	NAME	EXPERIENCE
1	Samruddhi	12
2	Aditi	7
3	Soham	9
4	Nikita	11

Under the Result Grid, the Output pane shows the Action Output:

#	Time	Action	Message
1	00:35:49	rename table Emp1 to Emp3	0 row(s) affected
2	00:35:49	select *from Emp3 LIMIT 0, 50000	4 row(s) returned

rename table Emp3 to Emp1;

select *from Emp1;

The screenshot shows a SQL query window with the following content:

```
60 -- truncate command on view--
61 • truncate table Emp1;
62
63 -- Rename command in view--
64 • rename table Emp1 to Emp3;
65 • select *from Emp3;
66 • rename table Emp3 to Emp1;
67 • select *from Emp1;
68
69 -- DML Commands On View (insert,update,delete)--
70
```

Below the code, there is a Result Grid showing the following data:

ID	NAME	EXPERIENCE
1	Samruddhi	12
2	Aditi	7
3	Soham	9
4	Nikita	11

Under the Result Grid, the Output pane shows the Action Output:

#	Time	Action	Message
1	00:35:49	rename table Emp1 to Emp3	0 row(s) affected
2	00:35:49	select *from Emp3 LIMIT 0, 50000	4 row(s) returned
3	00:36:18	rename table Emp3 to Emp1	0 row(s) affected
4	00:36:18	select *from Emp1 LIMIT 0, 50000	4 row(s) returned

DML commands on view(insert,update,delete);

Insert command:

```
insert into Emp1viewB values(6,'Prachi',21);
select *from Emp1viewB;
```

The screenshot shows the SQL Server Management Studio interface. At the top, there are tabs for 'create database ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 10*', 'SQL File 9*', and 'SQL File 11*'. Below the tabs, a code editor window displays the following SQL script:

```
65 • select *from Emp;
66 • rename table Emp3 to Emp1;
67 • select *from Emp1;
68
69 -- DML Commands On View (insert,update,delete)--
70 -- Insert command on view--
71 • insert into Emp1viewB values(6,'Prachi',21);
72 • select *from Emp1viewB;
73
74 -- update command on view--
```

Below the code editor is a 'Result Grid' table with three columns: ID, NAME, and EXPERIENCE. The data is as follows:

ID	NAME	EXPERIENCE
1	Samruddhi	12
4	Nikita	11
6	Prachi	21

At the bottom of the interface, there is an 'Output' window titled 'Emp1viewB 55'. It contains an 'Action Output' table with the following logs:

#	Time	Action	Message
1	00:35:49	rename table Emp1 to Emp3	0 row(s) affected
2	00:35:49	select *from Emp3 LIMIT 0, 50000	4 row(s) returned
3	00:36:18	rename table Emp3 to Emp1	0 row(s) affected
4	00:36:18	select *from Emp1 LIMIT 0, 50000	4 row(s) returned
5	00:42:07	insert into Emp1viewB values(6,'Prachi',21)	1 row(s) affected
6	00:42:07	select *from Emp1viewB LIMIT 0, 50000	3 row(s) returned

Update command:

```
update Emp1viewB set NAME='Kangude' where ID=1;  
select *from Emp1viewB;
```

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes options like '(create database)...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 10*', 'SQL File 9*', 'SQL File 11*', and 'SQLA'. Below the menu is a toolbar with various icons. The main window displays a script pane with numbered SQL statements. Statements 75 and 76 are highlighted, representing the update and select commands respectively. The result grid below shows a table with columns 'ID', 'NAME', and 'EXPERIENCE'. The final output pane at the bottom shows the execution log with 8 actions, including the update command and its results.

ID	NAME	EXPERIENCE
1	Kangude	12
4	Nikita	11
6	Prachi	21

Output:

#	Time	Action	Message
3	00:36:18	rename table Emp3 to Emp1	0 row(s) affected
4	00:36:18	select *from Emp1 LIMIT 0, 50000	4 row(s) returned
5	00:42:07	insert into Emp1viewB values(6,'Prachi',21)	1 row(s) affected
6	00:42:07	select *from Emp1viewB LIMIT 0, 50000	3 row(s) returned
7	00:42:41	update Emp1viewB set NAME='Kangude' where ID=1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
8	00:42:41	select *from Emp1viewB LIMIT 0, 50000	3 row(s) returned

Delete command:

delete from Emp1viewB where ID=5;
select *from Emp1viewB;

The screenshot shows the SQL Server Management Studio interface. In the top window, a query is being run against a database named 'Bank_Management_System'. The code includes several statements: a select statement, an update command, another select statement, a comment, a delete command (which is highlighted in blue), and a final select statement. The results grid below shows three rows of data: ID 1 with NAME 'Kangude' and EXPERIENCE 12, ID 4 with NAME 'Nikita' and EXPERIENCE 11, and ID 6 with NAME 'Prachi' and EXPERIENCE 21. In the bottom window, titled 'Emp1viewB 57', the 'Action Output' tab is selected, displaying a log of the executed actions with their times, descriptions, and messages. The log shows the insertion of a new row, selecting all rows, updating the name of the employee with ID 1, selecting all rows again, deleting the employee with ID 5, and finally selecting all rows again.

ID	NAME	EXPERIENCE
1	Kangude	12
4	Nikita	11
6	Prachi	21

#	Time	Action	Message
5	00:42:07	insert into Emp1viewB values(6,'Prachi',21)	1 row(s) affected
6	00:42:07	select *from Emp1viewB LIMIT 0, 50000	3 row(s) returned
7	00:42:41	update Emp1viewB set NAME='Kangude' where ID=1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
8	00:42:41	select *from Emp1viewB LIMIT 0, 50000	3 row(s) returned
9	00:43:09	delete from Emp1viewB where ID=5	0 row(s) affected
10	00:43:09	select *from Emp1viewB LIMIT 0, 50000	3 row(s) returned

Update the base table using its corresponding view restrictions on updatable views and perform view creation from multiple tables

update Emp1viewA set NAME='Yewale' where ID=6;
select *from Emp1viewA;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes 'create database ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 10*', 'SQL File 9*', and 'SQL File 11*' tabs. The 'SQL File 11*' tab is active. The code editor contains the following SQL script:

```
82 -- Update the base table using its corresponding view --
83 • update Emp1viewA set NAME='Yewale' where ID=6;
84 • select *from Emp1viewA;
85 • Select *from Emp1;
86
87 • update Emp2viewB set COUNTRY='AMERICA' WHERE ID=3;
88 • select *from Emp2viewB;
89 • select *from Emp2;
90
91
```

The 'Result Grid' pane displays a table with columns 'ID' and 'NAME'. The data is:

ID	NAME
1	Kangude
2	Aditi
3	Soham
4	Nikita
6	Yewale

The 'Output' pane shows the execution log:

#	Time	Action	Message
1	00:55:03	update Emp1viewA set NAME='Yewale' where ID=6	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	00:55:03	select *from Emp1viewA LIMIT 0, 50000	5 row(s) returned

Select *from Emp1;

The screenshot shows the SQL Server Management Studio interface. The top menu bar includes 'create database ...', 'SQL File 5*', 'Bank_Management_System', 'SQL File 5*', 'Food_Factory', 'SQL File 7', 'SQL File 8*', 'SQL File 10*', 'SQL File 9*', and 'SQL File 11*' tabs. The 'SQL File 11*' tab is active. The code editor contains the same SQL script as the previous screenshot:

```
81
82 -- Update the base table using its corresponding view --
83 • update Emp1viewA set NAME='Yewale' where ID=6;
84 • select *from Emp1viewA;
85 • Select *from Emp1;
86
87 • update Emp2viewB set COUNTRY='AMERICA' WHERE ID=3;
88 • select *from Emp2viewB;
89 • select *from Emp2;
90
91
```

The 'Result Grid' pane displays a table with columns 'ID', 'NAME', and 'EXPERIENCE'. The data is:

ID	NAME	EXPERIENCE
1	Kangude	12
2	Aditi	7
3	Soham	9
4	Nikita	11
6	Yewale	21

The 'Output' pane shows the execution log:

#	Time	Action	Message
1	00:55:03	update Emp1viewA set NAME='Yewale' where ID=6	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	00:55:03	select *from Emp1viewA LIMIT 0, 50000	5 row(s) returned
3	00:55:33	Select *from Emp1 LIMIT 0, 50000	5 row(s) returned

```
update Emp2viewB set COUNTRY='AMERICA' WHERE ID=3;
select *from Emp2viewB;
```

Screenshot of SQL Server Management Studio (SSMS) showing the execution of the provided SQL script.

The script updates the base table using its corresponding view:

```
81
82    -- Update the base table using its corresponding view --
83 • update Emp1viewA set NAME='Yewale' where ID=6;
84 • select *from Emp1viewA;
85 • Select *from Emp1;
86
87 • update Emp2viewB set COUNTRY='AMERICA' WHERE ID=3;
88 • select *from Emp2viewB;
89 • select *from Emp2;
90
^
```

The result grid shows the updated data:

ID	COUNTRY
3	AMERICA

The output pane shows the execution log:

#	Time	Action	Message
1	00:55:03	update Emp1viewA set NAME='Yewale' where ID=6	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	00:55:03	select *from Emp1viewA LIMIT 0, 50000	5 row(s) returned
3	00:55:33	Select *from Emp1 LIMIT 0, 50000	5 row(s) returned
4	00:56:21	update Emp2viewB set COUNTRY='AMERICA' WHERE ID=3	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
5	00:56:21	select *from Emp2viewB LIMIT 0, 50000	1 row(s) returned

```
select *from Emp2;
```

Screenshot of SQL Server Management Studio (SSMS) showing the execution of the provided SQL script.

The script updates the base table using its corresponding view:

```
81
82    -- Update the base table using its corresponding view --
83 • update Emp1viewA set NAME='Yewale' where ID=6;
84 • select *from Emp1viewA;
85 • Select *from Emp1;
86
87 • update Emp2viewB set COUNTRY='AMERICA' WHERE ID=3;
88 • select *from Emp2viewB;
89 • select *from Emp2;
90
^
```

The result grid shows the updated data:

ID	COUNTRY
1	India
2	USA
3	AMERICA

The output pane shows the execution log:

#	Time	Action	Message
1	00:55:03	update Emp1viewA set NAME='Yewale' where ID=6	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	00:55:03	select *from Emp1viewA LIMIT 0, 50000	5 row(s) returned
3	00:55:33	Select *from Emp1 LIMIT 0, 50000	5 row(s) returned
4	00:56:21	update Emp2viewB set COUNTRY='AMERICA' WHERE ID=3	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
5	00:56:21	select *from Emp2viewB LIMIT 0, 50000	1 row(s) returned
6	00:56:50	select *from Emp2 LIMIT 0, 50000	3 row(s) returned

Group C: PL/SQL

1. Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.

MySQL Stored procedure:

Name: `new_procedure` The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DLL:

```
1 * CREATE PROCEDURE `GetAllCustomers` ()  
2   BEGIN  
3     SELECT * FROM customers;  
4   END  
5
```

Apply **Revert**

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 USE `sakila`;  
2 DROP procedure IF EXISTS `sp_getCustomers`;  
3  
4 DELIMITER $$  
5 USE `sakila`$$  
6 CREATE PROCEDURE `sp_getCustomers` ()  
7 BEGIN  
8 select store_id, first_name, last_name, email, create_date, last_update  
9 from customer;  
10 END$$  
11  
12 DELIMITER ;  
13  
14
```

Back **Apply** **Cancel**

Screenshot of Oracle SQL Developer showing the Connections pane and a code editor window.

The Connections pane shows the following database structures:

- EDFSYS
- FLOW_FILES
- HR
 - Tables (Filtered)
 - Views
 - Editoring Views
 - Indexes
 - Packages
 - Procedures
 - ADD_JOB_HISTORY
 - SECURE_DML
 - Functions
 - Queues
 - Queues Tables
 - Triggers
 - Crossedition Triggers
 - Types
 - Sequences
 - Materialized Views
 - Materialized Views Logs
 - Synonyms
 - Database Links
 - Directories
 - Editions
 - Application Express
 - Java
 - XML Schemas
 - XML DB Repository
 - Recycle Bin
 - DBMS Jobs
- Scheduler

The code editor window displays the following PL/SQL procedure:

```
create or replace
PROCEDURE secure_dml
IS
BEGIN
  IF TO_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '16:00'
    OR TO_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
    RAISE_APPLICATION_ERROR (-20205,
      'You may only make changes during normal office hours');
  END IF;
END secure_dml;
```

The Running Log window shows the connection status:

```
Connecting to the database sandbox_as_sys.
Process exited.
Disconnecting from the database sandbox_as_sys.
```

The bottom status bar indicates "Messages" and "Running".

Screenshot of Oracle SQL Developer showing the new_procedure - Routine editor.

The routine details are as follows:

- Name: new_procedure
- DDL:

```
CREATE PROCEDURE `sp_getCustomers` ()
BEGIN
  select store_id, first_name, last_name, email, create_date, last_update
  from customer;
END
```

The bottom right corner contains "Apply" and "Revert" buttons, with "Apply" being highlighted by a red box.

PL/SQL Function:

function1 - Routine

Name: function1

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `function1`() RETURNS int(11)
2   BEGIN
3     declare var1 int;
4     declare var2 int;
5
6     select count(*) into var1 from table1;
7     set var2 = var1 + 1;
8
9     RETURN var2;
10    END
```

Routine

Apply Revert

stMySQL05* lbs_to_kg - Routine

Name: lbs_to_kg

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `lbs_to_kg`(lbs MEDIUMINT UNSIGNED) RETURNS varchar(50) CHARSET utf8mb4
2   DETERMINISTIC
3   SQL SECURITY INVOKER
4   COMMENT 'converts weight to kilograms and generates message'
5   BEGIN
6     DECLARE msg VARCHAR(50);
7     IF lbs > 999999 THEN SET msg =
8       CONCAT(ROUND((lbs * 0.45359237), 0), ' kg exceeds airport weight limits.');
9     ELSEIF lbs >= 100000 AND lbs <= 999999 THEN SET msg =
10      CONCAT(ROUND((lbs * 0.45359237), 0), ' kg exceeds runway weight limits.');
11    ELSE SET msg = CONCAT(ROUND((lbs * 0.45359237), 0), ' kg within weight limits.');
12  END IF;
13  RETURN msg;
14 END
```

Routine

Apply Revert

Worksheet | Query Builder

```
1 CREATE OR REPLACE FUNCTION get_total_sales(
2     in_year PLS_INTEGER
3 ) |
4 RETURN NUMBER
5 IS
6     l_total_sales NUMBER := 0;
7 BEGIN
8     -- get total sales
9     SELECT SUM(unit_price * quantity)
10    INTO l_total_sales
11   FROM order_items
12  INNER JOIN orders USING (order_id)
13 WHERE status = 'Shipped'
14 GROUP BY EXTRACT(YEAR FROM order_date)
15 HAVING EXTRACT(YEAR FROM order_date) = in_year;
16
17     -- return the total sales
18     RETURN l_total_sales;
19 END;
```

Group C: PL/SQL

2. Write and execute suitable database triggers. Consider row level and statement level triggers.

The screenshot shows the Oracle SQL Developer interface. In the top toolbar, there are icons for file operations, search, and help. Below the toolbar, a code editor displays the following PL/SQL code:

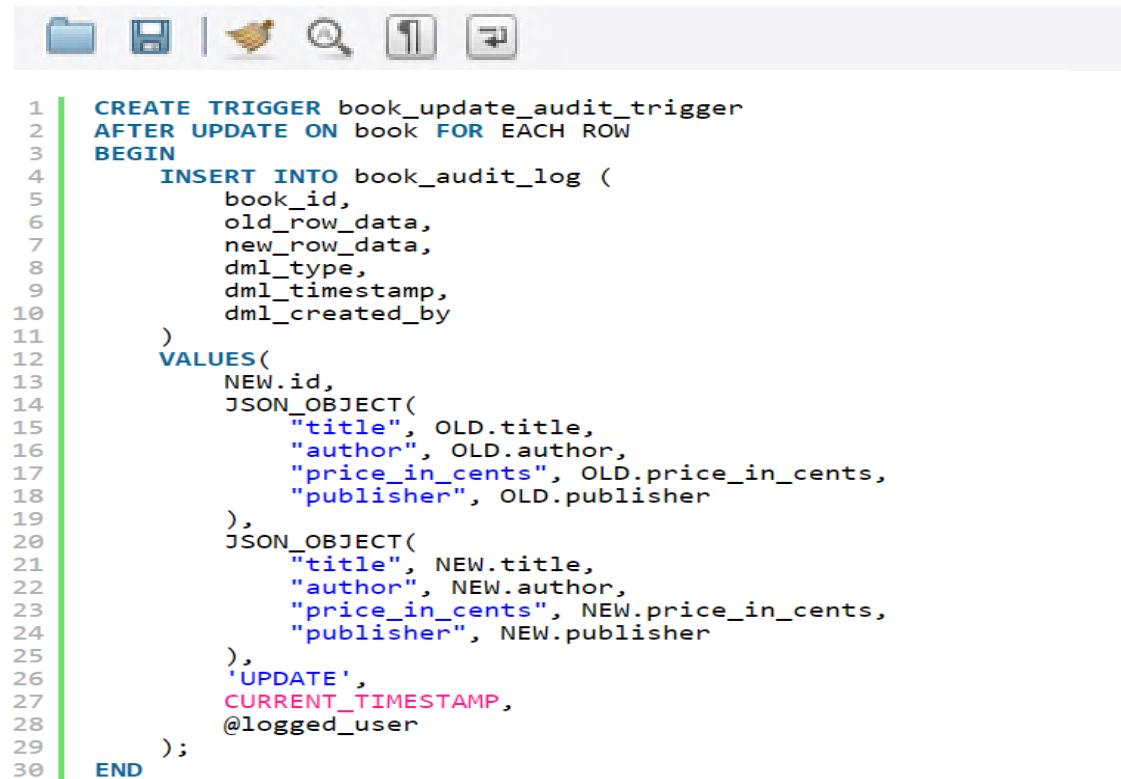
```
1 • CREATE TRIGGER login_trigger
2   BEFORE INSERT ON customers
3   FOR EACH ROW
4   INSERT INTO audit_log
5   SET audit_date = NOW(),
6       transaction_type = 'LOGIN',
7       account_id = NEW.customer_id;
8
```

The code is highlighted in blue, indicating it is valid PL/SQL. The 'login_trigger' line is bolded. The code editor has a light gray background with horizontal lines separating code lines. Below the code editor is an 'Output' panel with a title bar. The 'Action Output' tab is selected. It contains a table with the following data:| # | Time | Action | Message |
| --- | --- | --- | --- |
| 1 | 08:16:38 | CREATE TRIGGER login_trigger BEFORE INSERT ON custom... | 0 row(s) affected |

The screenshot shows the Oracle SQL Developer interface. In the top toolbar, there are icons for file operations, search, and help. Below the toolbar, a code editor displays the following PL/SQL code:

```
1 • USE `ga_abtest_logging`;
2 DELIMITER $$;
3 • CREATE TRIGGER avoid_empty
4   BEFORE INSERT ON daily_analytics
5   FOR EACH ROW
6   BEGIN
7     IF profileID = ''
8     THEN SET profileID = NULL;
9   END IF;
10  END$$
11
12
13
```

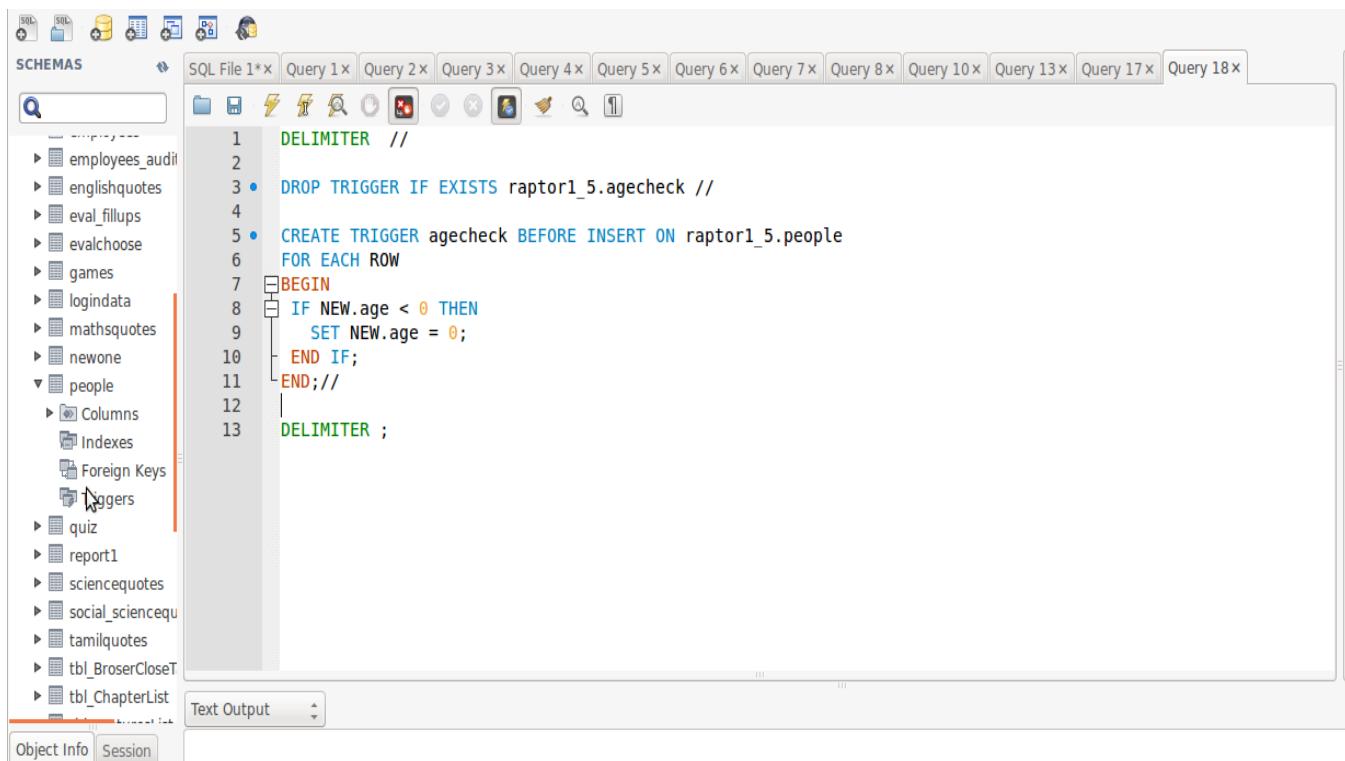
The code is highlighted in blue and orange. The 'avoid_empty' line is bolded. The code editor has a light gray background with horizontal lines separating code lines. The code is partially incomplete, ending with 'END\$\$'.



```

1 CREATE TRIGGER book_update_audit_trigger
2 AFTER UPDATE ON book FOR EACH ROW
3 BEGIN
4     INSERT INTO book_audit_log (
5         book_id,
6         old_row_data,
7         new_row_data,
8         dml_type,
9         dml_timestamp,
10        dml_created_by
11    )
12    VALUES(
13        NEW.id,
14        JSON_OBJECT(
15            "title", OLD.title,
16            "author", OLD.author,
17            "price_in_cents", OLD.price_in_cents,
18            "publisher", OLD.publisher
19        ),
20        JSON_OBJECT(
21            "title", NEW.title,
22            "author", NEW.author,
23            "price_in_cents", NEW.price_in_cents,
24            "publisher", NEW.publisher
25        ),
26        'UPDATE',
27        CURRENT_TIMESTAMP,
28        @logged_user
29    );
30 END

```



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

- Schemas:** A tree view of database schemas including employees_audit, englishquotes, eval_fillups, evalchoose, games, logindata, mathsquotes, newone, people (selected), quiz, report1, sciencequotes, social_sciencequ, tamilquotes, tbl_BroserCloseT, and tbl_ChapterList.
- SQL Editor:** Displays the following MySQL code:

```

1 DELIMITER //
2
3 • DROP TRIGGER IF EXISTS raptor1_5.agecheck //
4
5 • CREATE TRIGGER agecheck BEFORE INSERT ON raptor1_5.people
6   FOR EACH ROW
7   BEGIN
8       IF NEW.age < 0 THEN
9           SET NEW.age = 0;
10      END IF;
11  END//;
12
13 DELIMITER ;

```
- Object Info:** A tab at the bottom left of the interface.

Group C: PL/SQL

3. Write a PL/SQL block to implement all types of cursors.

```
create database slot;
use slot;
create table mytable(id int(2),name varchar(20),class varchar(20));
insert into mytable
values(1,'Samruddhi','Phd'),(2,'Aditi','Mtech'),(3,'Soham','Be'),(4,'Nikita','Btech'),(5,'Tejal','Diploma');
select *from mytable;
show tables;
drop database slot;

set @name_list = " ";
call mycur (@name_list);
select @name_list;
select *from mytable;
```

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

- Query Editor:** Contains the PL/SQL block provided in the question.
- Result Grid:** Displays the data from the mytable table:

id	name	class
1	Samruddhi	Phd
2	Aditi	Mtech
3	Soham	Be
4	Nikita	Btech
5	Tejal	Diploma
- Action Output:** Shows the log of actions taken:

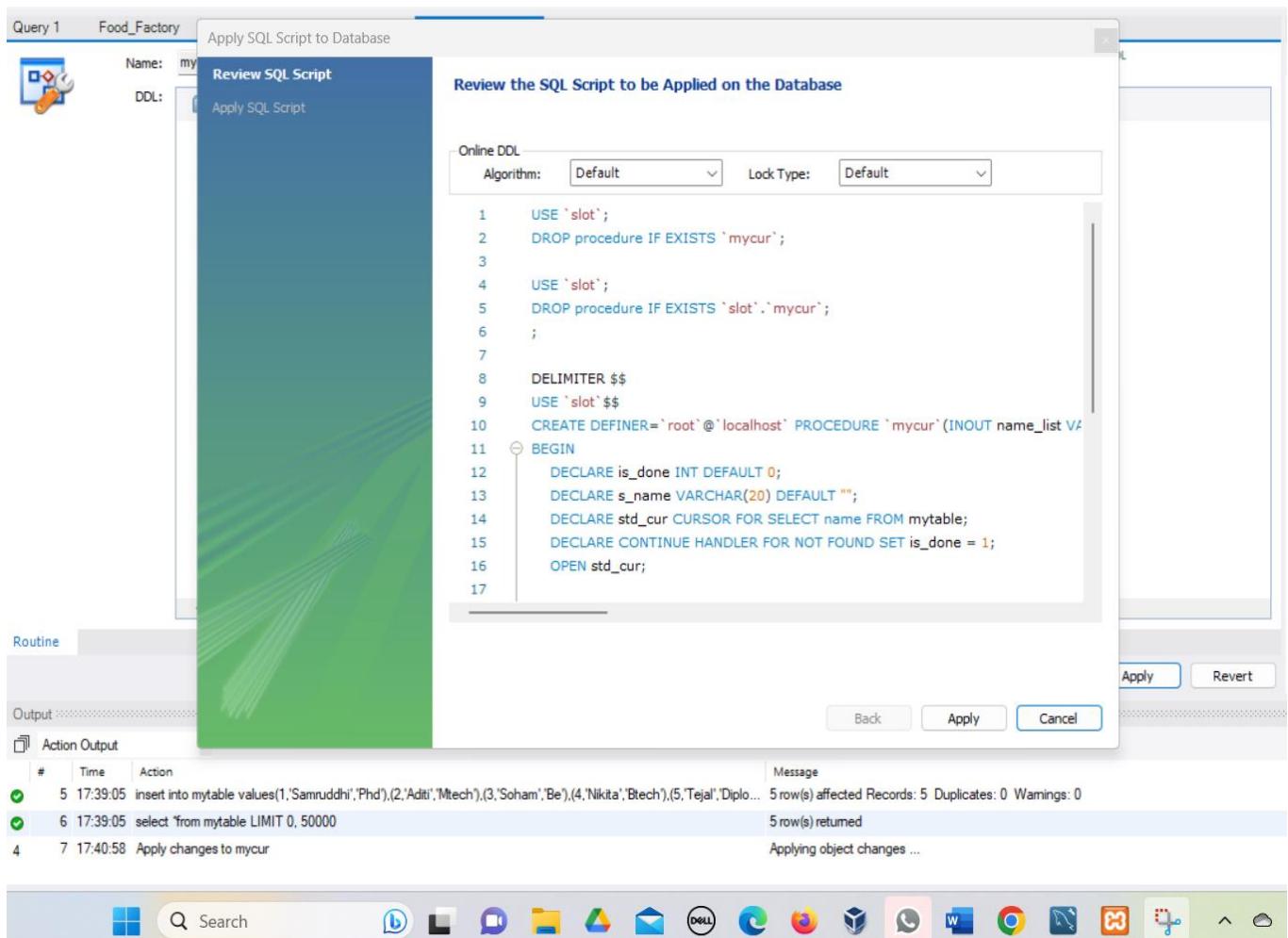
#	Time	Action	Message
4	17:39:05	show tables	1 row(s) returned
5	17:39:05	insert into mytable values(1,'Samruddhi','Phd'),(2,'Aditi','Mtech'),(3,'Soham','Be'),(4,'Nikita','Btech'),(5,'Tejal','Diploma')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
6	17:39:05	select *from mytable LIMIT 0, 50000	5 row(s) returned

CREATING STORED PROCEDURE IN CURSOR:

```
CREATE DEFINER='root'@'localhost' PROCEDURE `mycur`(INOUT name_list VARCHAR(100))
BEGIN
    DECLARE is_done INT DEFAULT 0;
    DECLARE s_name VARCHAR(20) DEFAULT "";
    DECLARE std_cur CURSOR FOR SELECT name FROM mytable;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET is_done = 1;
    OPEN std_cur;

    getdata: LOOP
        FETCH std_cur INTO s_name;
        IF is_done = 1 THEN
            LEAVE getdata;
        END IF;
        SET name_list = CONCAT(s_name, " ", name_list);
    END LOOP;

    CLOSE std_cur;
END
```



Name: mycur

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

```

1 • CREATE DEFINER='root'@'localhost' PROCEDURE `mycur` (INOUT name_list VARCHAR(100))
2 • BEGIN
3     DECLARE is_done INT DEFAULT 0;
4     DECLARE s_name VARCHAR(20) DEFAULT "";
5     DECLARE std_cur CURSOR FOR SELECT name FROM mytable;
6     DECLARE CONTINUE HANDLER FOR NOT FOUND SET is_done = 1;
7     OPEN std_cur;
8
9     getdata: LOOP
10        FETCH std_cur INTO s_name;
11        IF is_done = 1 THEN
12            LEAVE getdata;
13        END IF;
14        SET name_list = CONCAT(s_name, " ", name_list);
15    END LOOP;
16
17    CLOSE std_cur;
18 END

```

Routine

[Apply](#) [Revert](#)

Output

Action Output	Message
5 17:39:05 insert into mytable values(1,'Samruddhi','Phd'),(2,'Aditi','Mtech'),(3,'Soham','Be'),(4,'Nikita','Btech'),(5,'Tejal','Diploma')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
6 17:39:05 select *from mytable LIMIT 0, 50000	5 row(s) returned
7 17:40:58 Apply changes to mycur	Changes applied

```

set @name_list = " ";
call mycur (@name_list);
select @name_list;

```

Query 1 Food_Factory SQL File 5* mycur - Routine

```

1 create database slot;
2 • use slot;
3 • create table mytable(id int(2),name varchar(20),class varchar(20));
4 • show tables;
5 • insert into mytable values(1,'Samruddhi','Phd'),(2,'Aditi','Mtech'),(3,'Soham','Be'),(4,'Nikita','Btech'),(5,'Tejal','Diploma')
6 • select *from mytable;
7 • drop database slot;
8
9 • set @name_list = " ";
10 • call mycur (@name_list);
11 • select @name_list;
12 • select *from mytable;

```

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

@name_list
Tejal Nikita Soham Aditi Samruddhi

Result 10 x

[Read On](#)

Output

Action Output	Message
8 17:47:07 set @name_list = " "	0 row(s) affected
9 17:47:12 call mycur (@name_list)	0 row(s) affected
10 17:47:17 select @name_list LIMIT 0, 50000	1 row(s) returned

```
select *from mytable;
```

Query 1 Food_Factory SQL File 5* mycur - Routine

```
1 create database slot;
2 • use slot;
3 • create table mytable(id int(2),name varchar(20),class varchar(20));
4 • show tables;
5 • insert into mytable values(1,'Samruddhi','Phd'),(2,'Aditi','Mtech'),(3,'Soham','Be'),(4,'Nikita','Btech'),(5,'Tejal','Diploma');
6 • select *from mytable;
7 • drop database slot;
8
9 • set @name_list = " ";
10 • call mycur (@name_list);
11 • select @name_list;
12 • select *from mytable;
```

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

ID	Name	Class
1	Samruddhi	Phd
2	Aditi	Mtech
3	Soham	Be
4	Nikita	Btech
5	Tejal	Diploma

mytable 11 x Read On

Output

Action Output

#	Time	Action	Message
9	17:47:12	call mycur (@name_list)	0 row(s) affected
10	17:47:17	select @name_list LIMIT 0, 50000	1 row(s) returned
11	17:49:13	select *from mytable LIMIT 0, 50000	5 row(s) returned

Search

Group D: Relational Database Design

1. Design and case study of any organization (back end only).

Food Factory Project(back-end)

```
create database Food_Factory;  
use Food_Factory;
```

-- EMPLOYEE TABLE

use Food Factory;

```
create table Employee(
```

```
ID int not null primary key auto_increment primary key,Name_ varchar (50) not null,Address varchar (55) not null,Designation double not null,Salary double not null,DOJ date not null,Absents int not null,Received_Salary double not null);
```

```
show tables;
```

```
insert into Employee(Name ,Address,Designation,Salary,DOJ,Absents,Received Salary)
```

Kangude', 'Kothrud', 1, 30500, '2022-10-

04',5,((Employee.Designation*Employee.Salary)/(30))*(30-Employee.Absents)),

('Aditi Magar','Osmanabad',2,40500,'2022-02-04',1,((Employee.Designation*Employee.Salary)/(30))*(30-Employee.Absents)),

('Soham Pawar','Baramati',3,50500,'2022-02-02',2,((Employee.Designation*Employee.Salary)/(30))*(30-

(Employee.Absents)),
(Vaishnavi Gaikwad' 'Nashik' 5 60500 '2022-10-07' 3 ((Employee.Designation*Employee.Salary)/(30))*(30-

(Vaishnavi Employee.Absents)),

('Manthan Vaidya','Chakan',4,80500,'2022-06-10',4,((Employee.Designation*Employee.Salary)/(30))*(30-

(Employee.Absents)),
(Vikrant Kothimbire' 'Pune' 10 70500 '2022-10-06' 6 ((Employee Designation*Employee Salary)/(30))*(30-

(Vikrant Kothimore, Pune, 411050, 2022-10-06, 6, ((Employee.Designation * Employee.Salary) / 30)) * (30 - Employee.Absents)),
(Vishal Narwade, 'Karve Nagar', 411050, 2022-11-12, 8, ((Employee.Designation * Employee.Salary) / 30)) * (30 -

(Vishal Narwade, Karve Nagar, 8,90500, 2022-11-12, 8, ((Employee.Designation * Employee.Salary) / (30)) * (30 - Employee.Absents)),
(Siddharth choudhary, 'Mumbai', 9, 37500, '2022-12-11', 7, ((Employee.Designation * Employee.Salary) / (30)) * (30 -

(Vaishnavi Panchal' 'Latur' 677500 '2022-11-10' ((Employee.Designation*Employee.Salary)/(30))*(30-
Employee.Absents)),

(Vaishnavi Panchal ,Latur ,8, //500, 2022-11-10,10,((Employee.Designation*Employee.Salary)/(30))+(30-
Employee.Absents)),

((Employee.Designation*Employee.Salary)/(30))*(30-Employee.Absents));

```
Empty Statement;  
select *from Employee;
```

-- CUSTOMER TABLE

```

use Food_Factory;
create table Customer(
ID int not null auto_increment,Name_ varchar(50) not null,Address varchar(50) not null,Purchase_Items varchar(50),
Quantity int not null,Phone_Number varchar(50) not null,Product_Amount double not null,Pay_Amount double not
null,Date_date,Remaining_Amount int,PRIMARY KEY (ID));
show tables;
insert into
Customer(Name_,Address,Purchase_Items,Quantity,Phone_Number,Product_Amount,Pay_Amount,Date_,Remainin
g_Amount)
values('Dhanashri choudhary','Parner','2 Burger and 5 Bread',7,'9876543564',
Customer.Quantity*50,300,'2022-1-15',Customer.Product_Amount-Customer.Pay_Amount),
('Vaishnavi Andhale','Ahmadnagar','3 Burger and 6 Bread',9,'9876546564',
Customer.Quantity*50,200,'2022-2-15',Customer.Product_Amount-Customer.Pay_Amount),
('Prachi Yewale','wafgaon','3 Burger and 3 Bread',6,'9976566564',
Customer.Quantity*50,300,'2022-3-14',Customer.Product_Amount-Customer.Pay_Amount),
('Lalita Kshirsagar','Saswad','2 Burger and 2 Bread',4,'6767643564',
Customer.Quantity*50,200,'2022-4-5',Customer.Product_Amount-Customer.Pay_Amount),
('Tejal Sawale','Mawal','5 Burger and 5 Bread',10,'5555543564',
Customer.Quantity*50,100,'2022-6-15',Customer.Product_Amount-Customer.Pay_Amount),
('Divya Gharate','Nashik','9 Burger and 5 Bread',14,'9876543434',
Customer.Quantity*50,250,'2022-7-5',Customer.Product_Amount-Customer.Pay_Amount),
('Nikita Ghadage','mumbai','2 Burger and 10 Bread',12,'8877743564',
Customer.Quantity*50,150,'2022-8-25',Customer.Product_Amount-Customer.Pay_Amount),
('Arati Shinde','Khed','5 Burger and 5 Bread',10,'9866643564',
Customer.Quantity*50,100,'2022-8-10',Customer.Product_Amount-Customer.Pay_Amount),
('Nikita Dhulgande','Latur','4 Burger and 5 Bread',9,'7776543564',
Customer.Quantity*50,200,'2022-9-15',Customer.Product_Amount-Customer.Pay_Amount),
('Sakshi Ghanwat','Khed','4 Burger and 4 Bread',8,'8888543564',
Customer.Quantity*50,170,'2022-10-15',Customer.Product_Amount-Customer.Pay_Amount);
select *from Customer;

```

-- PURCHASE TABLE

Use food factory:

```
create table purchase (
```

```
create table purchase (
Serial_No int not null auto_increment,Salad double,Chocolate double not null,Bread double not null,PawBhaji double not null,MilkShake double not null,Pizza double not null,Sandwitch double not null,Cake double,Noodles double,Pasta double,Dosa double,Ice_Cream double,Burger double,Veg_Biryani double,sum double,PRIMARY KEY(Serial_No));
show tables;
```

insert

into

purchase(Salad,Chocolate,Bread,PawBhaji,MilkShake,Pizza,Sandwitch,Cake,Noodles,Pasta,Dosa,Ice_Cream,Burger ,Veg_Biryani,sum)

(3,5,25,7,7,2,4,5,6,2,7,2,9,8,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.MilkShake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),
 (4,5,16,4,6,3,4,6,8,9,2,7,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.MilkShake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),

(4,5,16,4,6,3,4,4,6,8,9,2,7,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.MilkShake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),
(5,5,26,7,6,3,5,2,3,2,2,7,4,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.MilkShake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),

(3,5,7,5,6,2,2,2,2,7,4,7,2,3,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.Milk Shake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),
 (4,3,14,7,6,25,2,7,2,2,5,2,8,2 purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.Milk Shake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),

(4,5,14,7,6,23,2,1,2,2,5,2,8,2,purchasc.Salad+purchasc.Chocolate+purchase.BTCad+purchase.FawBhaji+purchase.MilkShake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),

(2,5,6,7,6,6,7,5,2,9,2,2,7,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.Milk Shake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),
 (8,5,2,7,6,7,4,2,3,5,7,2,2,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.Milk

(8,5,5,7,6,7,4,2,5,5,7,2,2,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.Milk Shake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani),
(9,5,6,3,6,2,3,2,8,2,9,5,2,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.Milk

(6,7,26,7,3,7,2,7,2,9,9,9,5,2,purchase.Salad+purchase.Chocolate+purchase.Bread+purchase.PawBhaji+purchase.Mil

kShake+purchase.Pizza+purchase.Sandwitch+purchase.Cake+purchase.Noodles+purchase.Pasta+purchase.Dosa+purchase.Ice_Cream+purchase.Burger+purchase.Veg_Biryani);
select *from purchase;

select * from purchase;

	Serial_No	Salad	Chocolate	Bread	PawBhaji	MilkShake	Pizza	Sandwitch	Cake	Noodles	Pasta	Dosa	Ice_Cream	Burger	Veg_Biryani	sum
►	1	2	5	26	7	6	2	2	2	2	2	2	2	2	2	64
►	2	2	5	25	7	7	2	4	5	6	7	2	2	2	2	62

2	3	5	25	7	7	2	4	5	6	2	7	2	9	8	92
3	4	5	16	4	6	3	4	4	6	8	9	2	7	2	80
4	5	5	26	7	6	3	5	2	3	2	2	7	4	2	79
5	5	5	7	5	6	3	3	3	3	7	1	7	3	3	57

-- SALE MAN TABLE

```

use food_factory;
create table sale_man(ID int not null auto_increment,Name_ varchar (50)not null,Address varchar(50) not null,Purchase_Items varchar(50),Quantity int not null,Phone_Number varchar(50) not null,sum double not null,Pay_Amount double not null,Date_date,Remaining_Amount int,PRIMARY KEY (ID));
show tables;
insert into
sale_man(Name_,Address,Purchase_Items,Quantity,Phone_Number,sum,Pay_Amount,Date_,Remaining_Amount)
values('Nikita nagwade','parner','5 burgers and 10 breads',15,'9087698769',sale_man.Quantity*50,500,'2022-11-10',sale_man.sum-sale_man.Pay_Amount),
('Vaishnavi andhale','Ahmadnagar','6 burgers and 1 breads',7,'8087623143',sale_man.Quantity*50,200,'2022-03-12',sale_man.sum-sale_man.Pay_Amount),
('Dhanashri chaudhary','Nagar','7 burgers and 5 breads',13,'7087667654',sale_man.Quantity*50,390,'2022-07-11',sale_man.sum-sale_man.Pay_Amount),
('Nikita ghadge','Mumbai','4 burgers and 7 breads',11,'6087654328',sale_man.Quantity*50,450,'2022-11-12',sale_man.sum-sale_man.Pay_Amount),
('Sakshi ghanwat','Khed','3 burgers and 8 breads',11,'5087690909',sale_man.Quantity*50,300,'2022-12-09',sale_man.sum-sale_man.Pay_Amount),
('Divya gharate','Nashik','2 burgers and 10 breads',12,'8889094923',sale_man.Quantity*50,150,'2022-10-10',sale_man.sum-sale_man.Pay_Amount),
('Tejal sawale','Mawal','9 burgers and 9 breads',18,'6765655320',sale_man.Quantity*50,250,'2022-04-11',sale_man.sum-sale_man.Pay_Amount),
('Nikita dhulgande','Latur','1 burgers and 5 breads',6,'7587878323',sale_man.Quantity*50,100,'2022-07-12',sale_man.sum-sale_man.Pay_Amount),
('Prachi yewale','Rajgurunagar','5 burgers and 4 breads',9,'8734343328',sale_man.Quantity*50,150,'2022-09-03',sale_man.sum-sale_man.Pay_Amount),
('Lalita kshirsagar','Saswad','8 burgers and 2 breads',10,'7088884323',sale_man.Quantity*50,100,'2022-10-02',sale_man.sum-sale_man.Pay_Amount);
select *from sale_man;

```

-- SALARIES TABLE

```
use food_factory;
create table salaries(serial_no int auto_increment primary key,employee_id int,employee_name
varchar(40),employee_salary int,foreign key(employee_id) references employee (id));
show tables;
insert into salaries(employee_id,employee_name,employee_salary)
values(1,(select name_ from employee where id=1),
(select received_salary from employee where id=1)),
(2,(select name_ from employee where id=2),
(select received_salary from employee where id=2)),
(3,(select name_ from employee where id=3),
(select received_salary from employee where id=3)),
(4,(select name_ from employee where id=4),
(select received_salary from employee where id=4)),
(5,(select name_ from employee where id=5),
(select received_salary from employee where id=5)),
(6,(select name_ from employee where id=6),
(select received_salary from employee where id=6)),
(7,(select name_ from employee where id=7),
(select received_salary from employee where id=7)),
(8,(select name_ from employee where id=8),
(select received_salary from employee where id=8)),
(9,(select name_ from employee where id=9),
(select received_salary from employee where id=9)),
(10,(select name_ from employee where id=10),
(select received_salary from employee where id=10));
select *from salaries;
```

	serial_no	employee_id	employee_name	employee_salary
▶	1	1	Samruddhi Kangude	25417
	2	2	Aditi Magar	78300
	3	3	Soham Pawar	141400
	4	4	Vaishnavi Gaikwad	272250
	5	5	Manthan Vaidya	279067
	6	6	Vikrant Kothimbire	564000
	7	7	Vishal Narwade	530933
	8	8	Siddharth choudhary	258750
	9	9	Vaishnavi Panchal	310000
	10	10	Nikita Nagwade	330750
*	NULL	NULL	NULL	NULL

-- EXPENSES TABLE

```
use food_factory;
create table expenses(
serial_no int primary key auto_increment,
purchase_product double,
renovation double,
salaries_double,
sum_of_expenses double,
date_date
);
show tables;
insert into expenses(purchase_product,renovation,salaries_,sum_of_expenses,date_ )
values((select sum(sum)from purchase),500,(select sum(employee_salary)from salaries),
expenses.purchase_product+expenses.renovation+expenses.salaries_,'2022-11-15');
select *from expenses;
```

Query 1 Food_Factory x

178 values((select sum(sum)from purchase),500,(select sum(employee_salary)from salaries),
179 expenses.purchase_product+expenses.renovation+expenses.salaries_,'2022-11-15');
180 • select *from expenses;
181
182 -- SALES TABLE
183 • use food_factory;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	serial_no	purchase_product	renovation	salaries_	sum_of_expenses	date_
▶	1	757	500	2790867	2792124	2022-11-15
*	HULL	HULL	NULL	HULL	NULL	HULL

Result 11 expenses 12 x

Output:

Action Output

#	Time	Action	Message
30	09:36:03	show tables	6 row(s) returned
31	09:36:03	insert into expenses(purchase_product,renovation,salaries_,sum_of_expenses,date_) values((select sum(sum)fr...	1 row(s) affected
32	09:36:03	select *from expenses LIMIT 0,50000	1 row(s) returned

Search

Dell

Firefox

Word

Excel

-- SALES TABLE

```
use food_factory;
create table sales(
serial_no int auto_increment primary key,
sale_man_sales int,
customer_sales int);
show tables;
insert into sales(sale_man_sales, customer_sales)
values((select sum from sale_man where id=1),(select product_Amount from customer where id=1)),
((select sum from sale_man where id=2),(select product_Amount from customer where id=2)),
((select sum from sale_man where id=3),(select product_Amount from customer where id=3)),
((select sum from sale_man where id=4),(select product_Amount from customer where id=4)),
((select sum from sale_man where id=5),(select product_Amount from customer where id=5)),
((select sum from sale_man where id=6),(select product_Amount from customer where id=6)),
((select sum from sale_man where id=7),(select product_Amount from customer where id=7)),
((select sum from sale_man where id=8),(select product_Amount from customer where id=8)),
((select sum from sale_man where id=9),(select product_Amount from customer where id=9)),
((select sum from sale_man where id=10),(select product_Amount from customer where id=10));
select *from sales;
```

Query 1 Food_Factory

```
198 ((select sum from sale_man where id=9),(select product_Amount from customer where id=9)),
199 ((select sum from sale_man where id=10),(select product_Amount from customer where id=10));
200 • select *from sales;
201
202 -- PROFIT TABLE
203 • use food_factory;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

serial_no	sale_man_sales	customer_sales
1	750	350
2	350	450
3	650	300
4	550	200
5	550	500
6	600	700
7	900	600
8	300	500
9	450	450
10	500	400
*	NULL	NULL

Result 13 sales 14

Action Output

#	Time	Action	Message
35	09:37:09	show tables	7 row(s) returned
36	09:37:09	insert into sales(sale_man_sales, customer_sales) values((select sum from sale_man where id=1),(select product_Amount from customer where id=1)),((select sum from sale_man where id=2),(select product_Amount from customer where id=2)),((select sum from sale_man where id=3),(select product_Amount from customer where id=3)),((select sum from sale_man where id=4),(select product_Amount from customer where id=4)),((select sum from sale_man where id=5),(select product_Amount from customer where id=5)),((select sum from sale_man where id=6),(select product_Amount from customer where id=6)),((select sum from sale_man where id=7),(select product_Amount from customer where id=7)),((select sum from sale_man where id=8),(select product_Amount from customer where id=8)),((select sum from sale_man where id=9),(select product_Amount from customer where id=9)),((select sum from sale_man where id=10),(select product_Amount from customer where id=10));	10 row(s) affected Records
37	09:37:09	select *from sales LIMIT 0, 50000	10 row(s) returned

Search

-- PROFIT TABLE

```
use food_factory;
create table profit(
day_int,
expenses double,
purchase double,
salary double,
daily_profit int,
foreign key(day_) references expenses (serial_no)
);
show tables;
insert into profit(expenses,purchase,salary,daily_profit)
values((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=1),
(select employee_salary from salaries where serial_no=1),
((select sale_man_sales from sales where serial_no=1) +
(select customer_sales from sales where serial_no=1))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=2),
(select employee_salary from salaries where serial_no=2),
((select sale_man_sales from sales where serial_no=2) +
(select customer_sales from sales where serial_no=2))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=3),
(select employee_salary from salaries where serial_no=3),
((select sale_man_sales from sales where serial_no=3) +
(select customer_sales from sales where serial_no=3))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=4),
(select employee_salary from salaries where serial_no=4),
((select sale_man_sales from sales where serial_no=4) +
(select customer_sales from sales where serial_no=4))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=5),
(select employee_salary from salaries where serial_no=5),
((select sale_man_sales from sales where serial_no=5) +
(select customer_sales from sales where serial_no=5))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=6),
(select employee_salary from salaries where serial_no=6),
((select sale_man_sales from sales where serial_no=6) +
(select customer_sales from sales where serial_no=6))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=7),
(select employee_salary from salaries where serial_no=7),
((select sale_man_sales from sales where serial_no=7) +
(select customer_sales from sales where serial_no=7))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=8),
(select employee_salary from salaries where serial_no=8),
((select sale_man_sales from sales where serial_no=8) +
(select customer_sales from sales where serial_no=8))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=9),
(select employee_salary from salaries where serial_no=9),
((select sale_man_sales from sales where serial_no=9) +
(select customer_sales from sales where serial_no=9))-(profit.expenses+profit.purchase+profit.salary)),
((select sum_of_expenses from expenses where serial_no=1),
(select sum from purchase where serial_no=10),
```

```
(select employee_salary from salaries where serial_no=10),
((select sale_man_sales from sales where serial_no=10)+  

(select customer_sales from sales where serial_no=10))-(profit.expenses+profit.purchase+profit.salary));  

select *from profit;
```

Query 1 Food_Factory

261 (select employee_salary from salaries where serial_no=10),
262 ((select sale_man_sales from sales where serial_no=10)+
263 (select customer_sales from sales where serial_no=10))-(profit.expenses+profit.purchase+profit.salary));
264 • select *from profit;
265
266 -- MENU TABLE

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

day_	expenses	purchase	salary	daily_profit
HULL	2792124	64	25417	-2816505
HULL	2792124	92	78300	-2869716
HULL	2792124	80	141400	-2932654
HULL	2792124	79	272250	-3063703
HULL	2792124	57	279067	-3070198
HULL	2792124	89	564000	-3354913
HULL	2792124	68	530933	-3321625
HULL	2792124	63	258750	-3050137
HULL	2792124	64	310000	-3101288
HULL	2792124	101	330750	-3122075

Result 15 profit 16

Action Output

#	Time	Action	Message
40	09:38:44	show tables	8 row(s) returned
41	09:38:44	insert into profit(expenses,purchase,salary,daily_profit) values((select sum_of_expenses from expenses where s...)	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
42	09:38:45	select *from profit LIMIT 0, 50000	10 row(s) returned

-- MENU TABLE

```
use food_factory;
create table menu(
serial_no int auto_increment primary key,
saleman_list varchar(50),
employee_list varchar(50),
customer_list varchar(50),
profit double,
product double not null,
salaries double not null);
insert into menu(saleman_list,employee_list,customer_list,profit,product,salaries)
values((select name_ from sale_man where id=1),
(select name_ from employee where id=1),
(select name_ from customer where id=1),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=1),
(select employee_salary from salaries where serial_no=1)),
((select name_ from sale_man where id=2),
(select name_ from employee where id=2),
(select name_ from customer where id=2),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=2),
(select employee_salary from salaries where serial_no=2)),
((select name_ from sale_man where id=3),
(select name_ from employee where id=3),
(select name_ from customer where id=3),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=3),
(select employee_salary from salaries where serial_no=3)),
((select name_ from sale_man where id=4),
(select name_ from employee where id=4),
(select name_ from customer where id=4),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=4),
(select employee_salary from salaries where serial_no=4)),
((select name_ from sale_man where id=5),
(select name_ from employee where id=5),
(select name_ from customer where id=5),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=5),
(select employee_salary from salaries where serial_no=5)),
((select name_ from sale_man where id=6),
(select name_ from employee where id=6),
(select name_ from customer where id=6),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=6),
(select employee_salary from salaries where serial_no=6)),
((select name_ from sale_man where id=7),
(select name_ from employee where id=7),
(select name_ from customer where id=7),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=7),
(select employee_salary from salaries where serial_no=7)),
((select name_ from sale_man where id=8),
(select name_ from employee where id=8),
(select name_ from customer where id=8),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=8),
(select employee_salary from salaries where serial_no=8)),
```

```
((select name_ from sale_man where id=9),
(select name_ from employee where id=9),
(select name_ from customer where id=9),
(select sum(daily_profit)from profit where day_=1),
(select sum from purchase where serial_no=9),
(select employee_salary from salaries where serial_no=9)),
((select name_ from sale_man where id=10),
(select name_ from employee where id=10),
(select name_ from customer where id=10),
(select sum(daily_profit)from profit where day_=10),
(select sum from purchase where serial_no=1),
(select employee_salary from salaries where serial_no=10));
select *from menu;
```

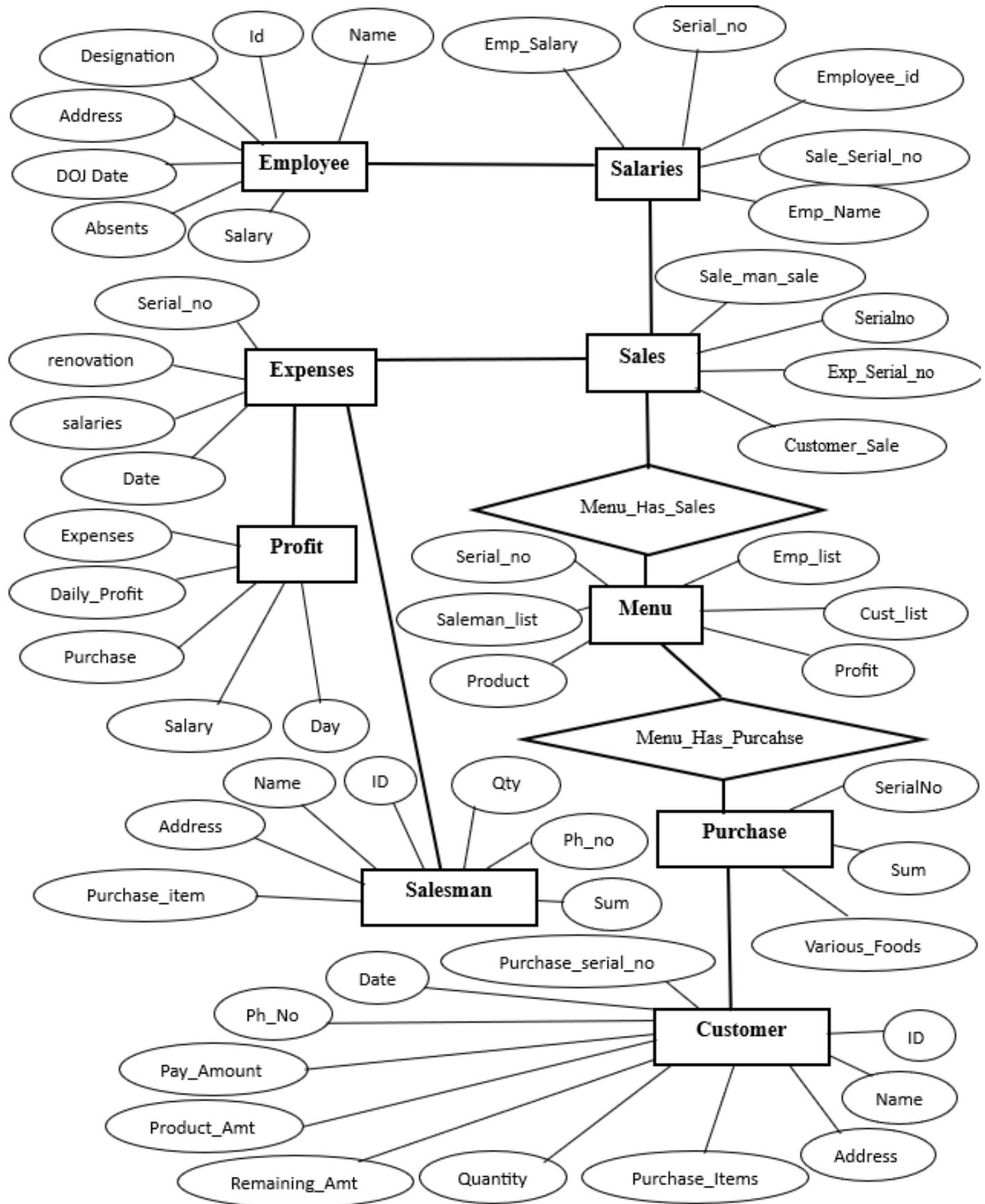
```
drop database Food_Factory;
```

Group D: Relational Database Design

Project ER Diagram and Database Design

For ER diagram and Database design following guidelines can be used:

Draw an ER diagram of your project.



- 2. Reduce this ER diagram into the tables and complete database design.**
- 3. Subsequently, list all the functional dependencies on each table that you expect will hold.**
- 4. Check that the database schema is in 3NF/BCNF. If it is not, apply normalization. Use non loss decomposition and bring the database schema in 3NF/BCNF.**
- Give the ER diagram and the data dictionary as part of the requirement specifications file which you created for the project proposal.**

