LAB – 1

Name: Gandevia Keval Dharmeshbhai

Sem: VII

Roll No: CE046

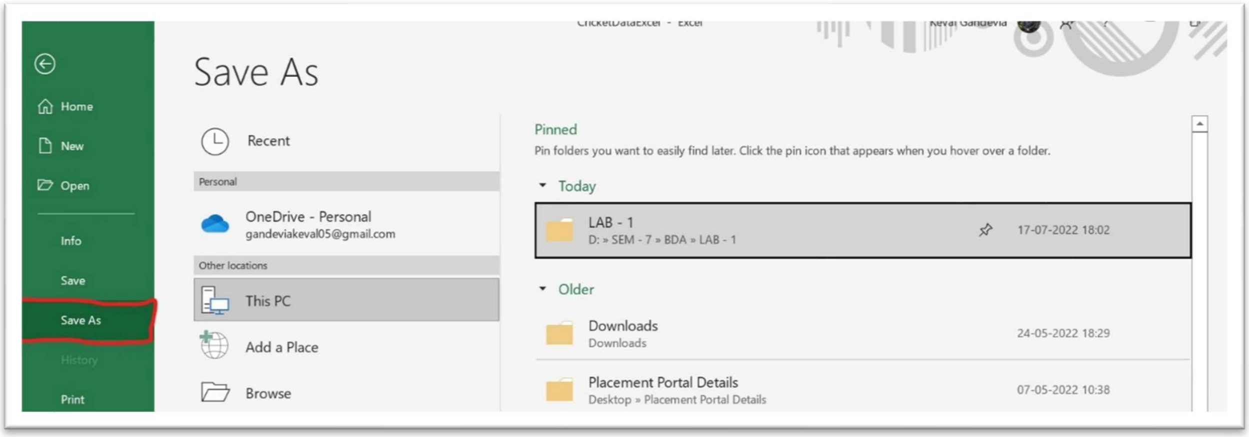
Subject: Big Data and Analytics

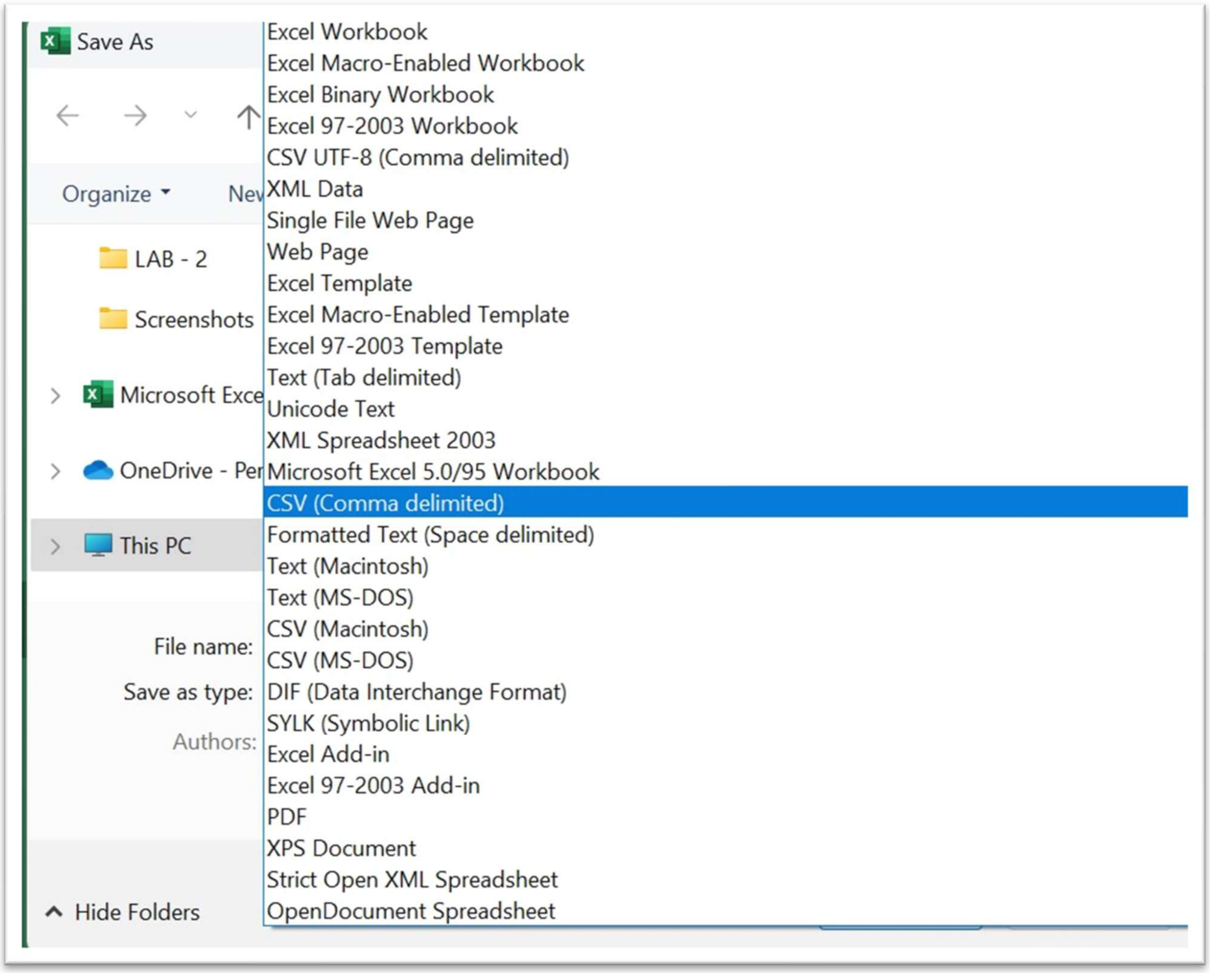
Aim: Recording types of data and various file formats. Identifying data sources. Handling traditionally to start with a small scale.

Q. 1: Given the spreadsheet file convert it into a csv.

❖ There are many ways to convert the spreadsheet file into csv.

I. Save As spreadsheet file with .csv extension.



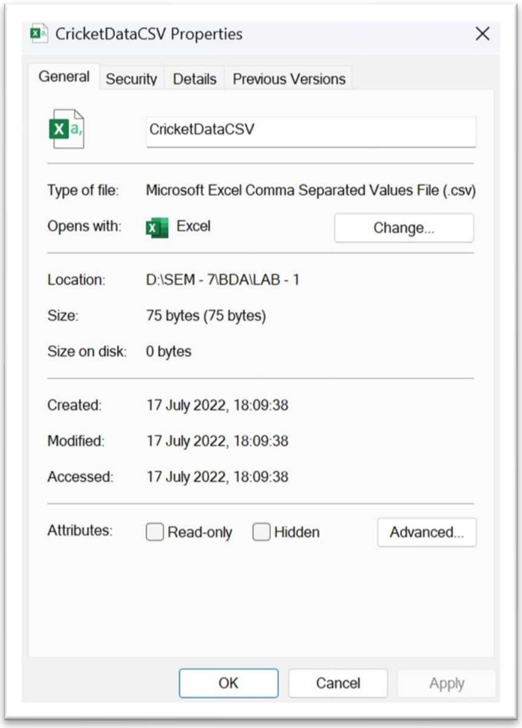


File name:

Save as type:

Authors:

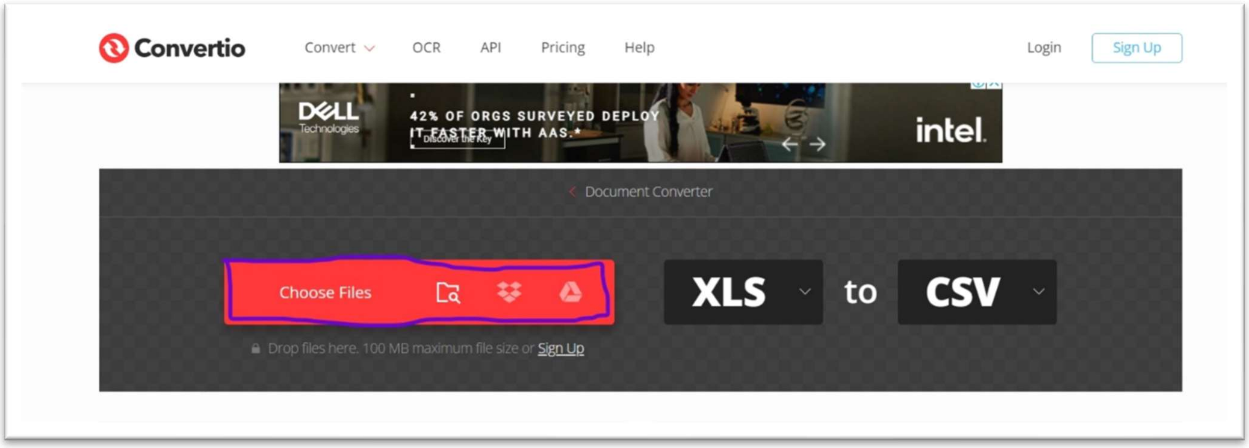
Excel Workbook
Excel Macro-Enabled Workbook
Excel Binary Workbook
Excel 97-2003 Workbook
CSV UTF-8 (Comma delimited)
XML Data
Single File Web Page
Web Page
Excel Template
Excel Macro-Enabled Template
Excel 97-2003 Template
Text (Tab delimited)
Unicode Text
XML Spreadsheet 2003
Microsoft Excel 5.0/95 Workbook
CSV (Comma delimited)
Formatted Text (Space delimited)
Text (Macintosh)
Text (MS-DOS)
CSV (Macintosh)
CSV (MS-DOS)
DIF (Data Interchange Format)
SYLK (Symbolic Link)
Excel Add-in
Excel 97-2003 Add-in
PDF
XPS Document
Strict Open XML Spreadsheet
OpenDocument Spreadsheet

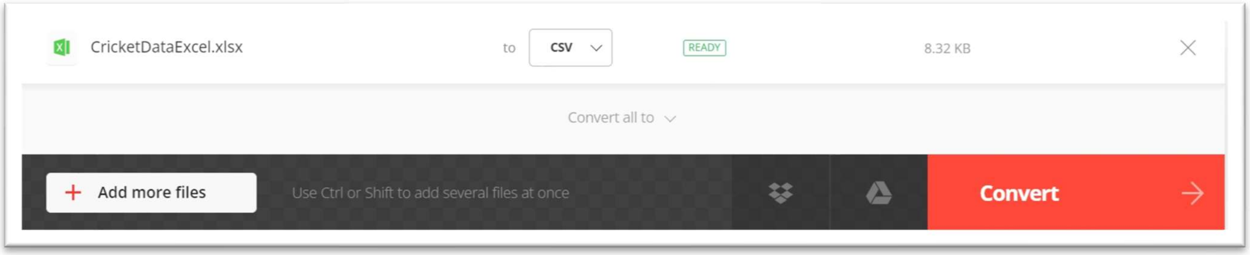


CricketDataCSV

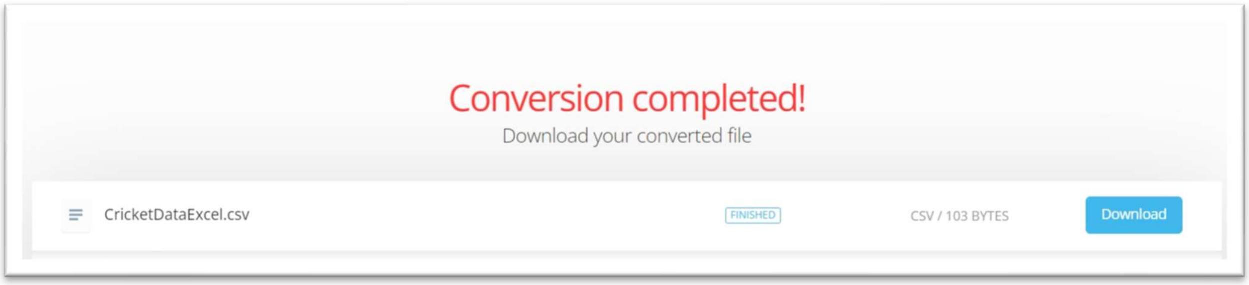
Type of file:	Microsoft Excel Comma Separated Values File (.csv)
Opens with:	Excel
Location:	D:\SEM - 7\BDA\LAB - 1
Size:	75 bytes (75 bytes)
Size on disk:	0 bytes
Created:	17 July 2022, 18:09:38
Modified:	17 July 2022, 18:09:38
Accessed:	17 July 2022, 18:09:38
Attributes:	☐ Read-only ☐ Hidden

II. **Use online converter.**

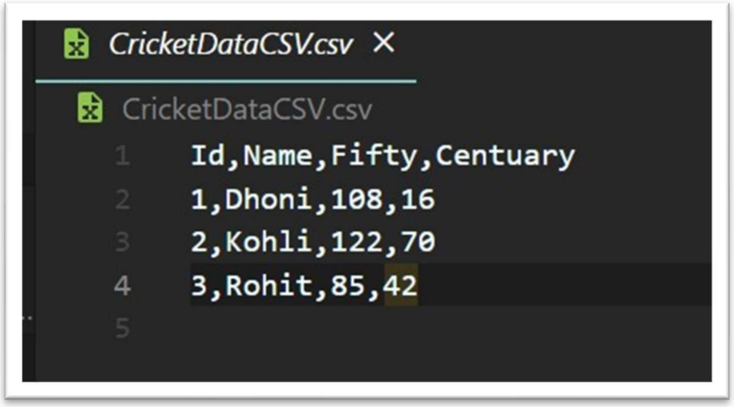




CricketDataExcel.xlsx to CSV READY 8.32 KB


Conversion completed!
Download your converted file

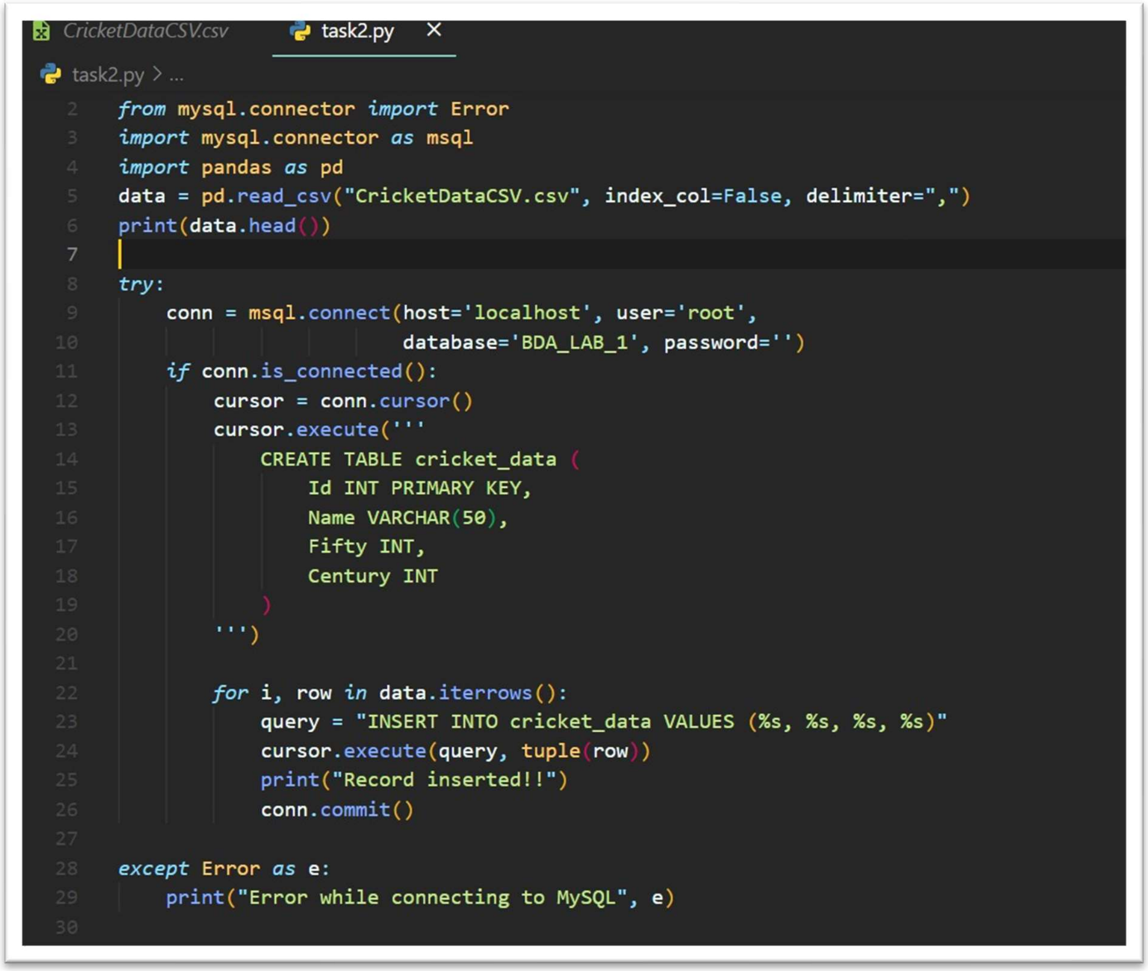
CricketDataExcel.csv FINISHED CSV / 103 BYTES Download


CricketDataCSV.csv

```
Id,Name,Fifty,Centuary
1,Dhoni,108,16
2,Kohli,122,70
3,Rohit,85,42
```

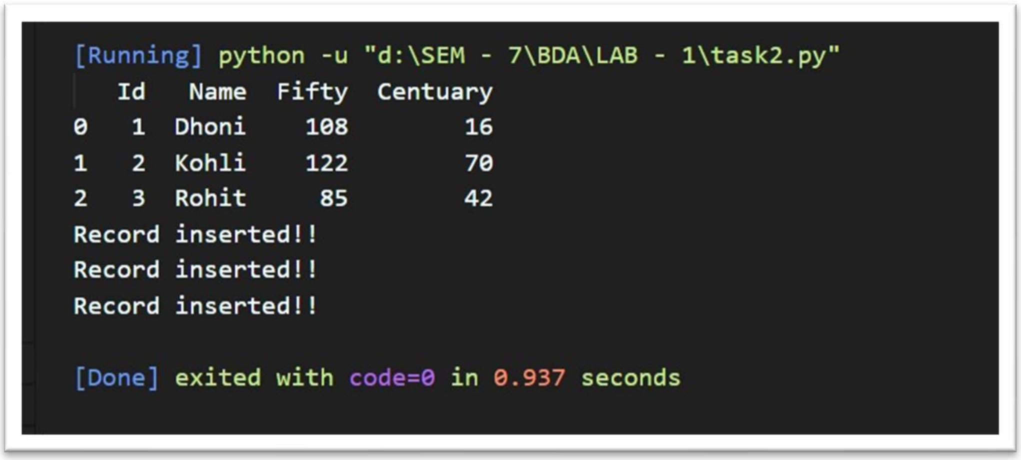
Q. 2: Import csv into MySQL database table.

❖ **Code:**

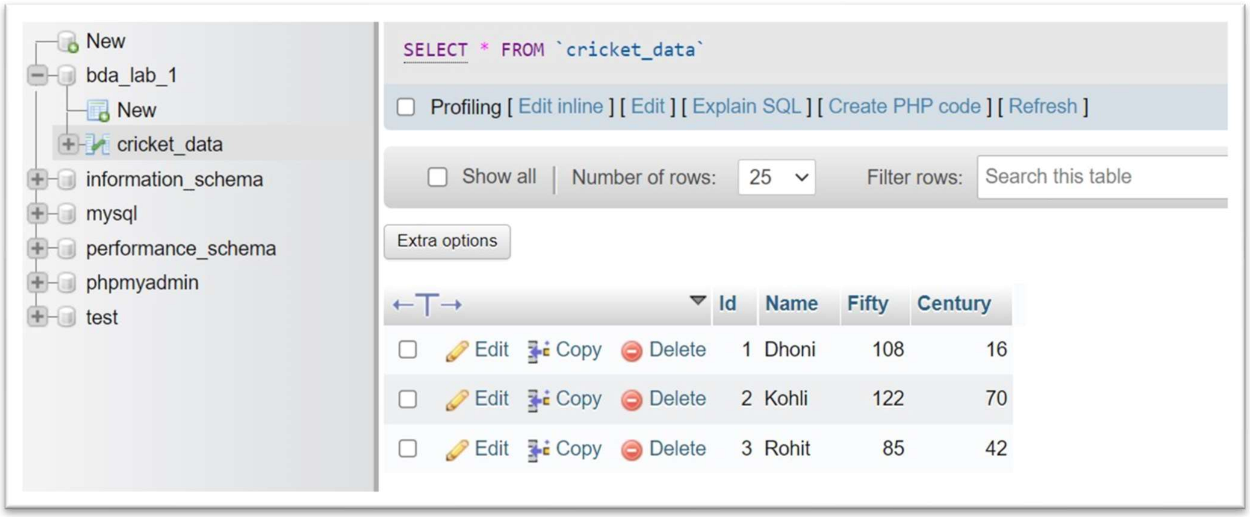


```python
from mysql.connector import Error
import mysql.connector as msql
import pandas as pd
data = pd.read_csv("CricketDataCSV.csv", index_col=False, delimiter=",")
print(data.head())

try:
    conn = msql.connect(host='localhost', user='root',
                        database='BDA_LAB_1', password='')
    if conn.is_connected():
        cursor = conn.cursor()
        cursor.execute('''
            CREATE TABLE cricket_data (
                Id INT PRIMARY KEY,
                Name VARCHAR(50),
                Fifty INT,
                Century INT
            )
        ''')

        for i, row in data.iterrows():
            query = "INSERT INTO cricket_data VALUES (%s, %s, %s, %s)"
            cursor.execute(query, tuple(row))
            print("Record inserted!!")
            conn.commit()

except Error as e:
    print("Error while connecting to MySQL", e)

```

❖ **Output:**



```
[Running] python -u "d:\SEM - 7\BDA\LAB - 1\task2.py"
   Id   Name  Fifty  Centuary
0   1  Dhoni    108        16
1   2  Kohli    122        70
2   3  Rohit     85        42
Record inserted!!
Record inserted!!
Record inserted!!

[Done] exited with code=0 in 0.937 seconds
```



New
bda_lab_1
New
cricket_data
information_schema
mysql
performance_schema
phpmyadmin
test

```
SELECT * FROM `cricket_data`
```

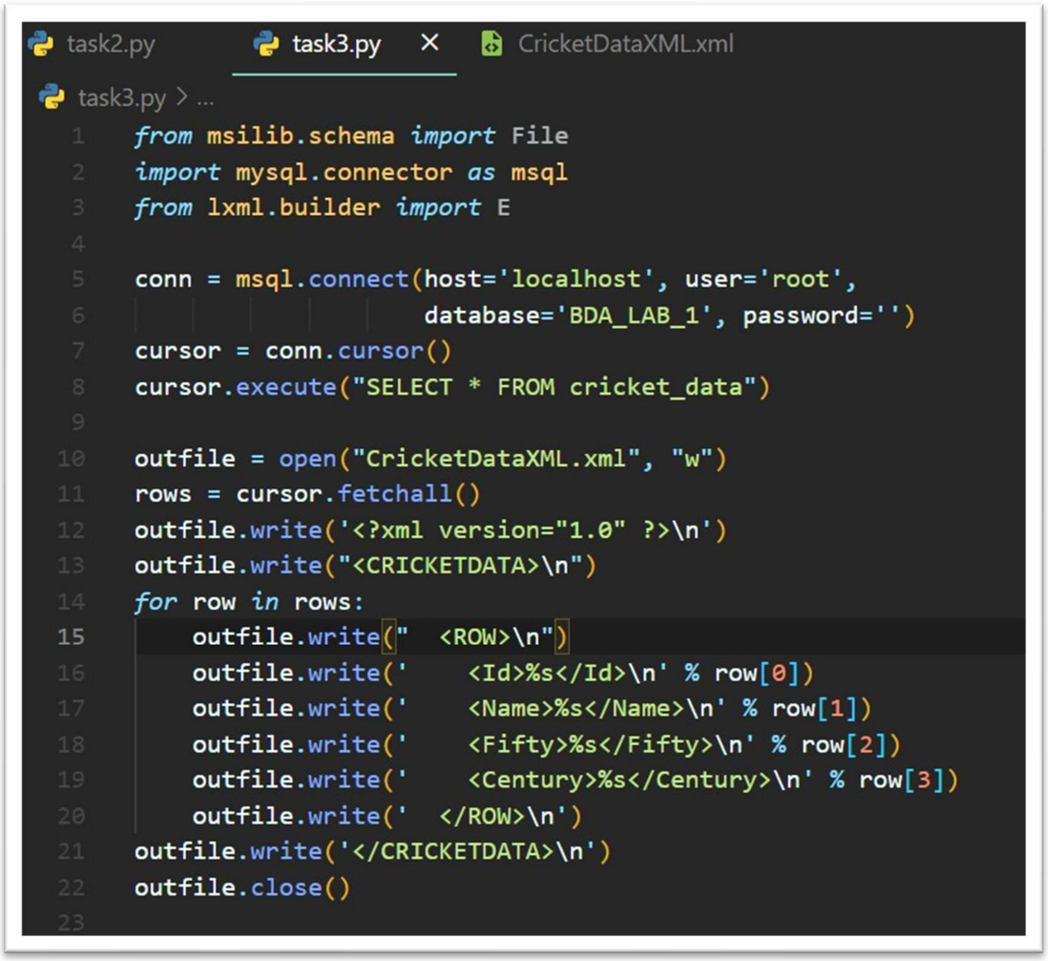
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

Id	Name	Fifty	Century
1	Dhoni	108	16
2	Kohli	122	70
3	Rohit	85	42

Q. 3: Write a computer program to read records from database and generate data file.

I. **XML**

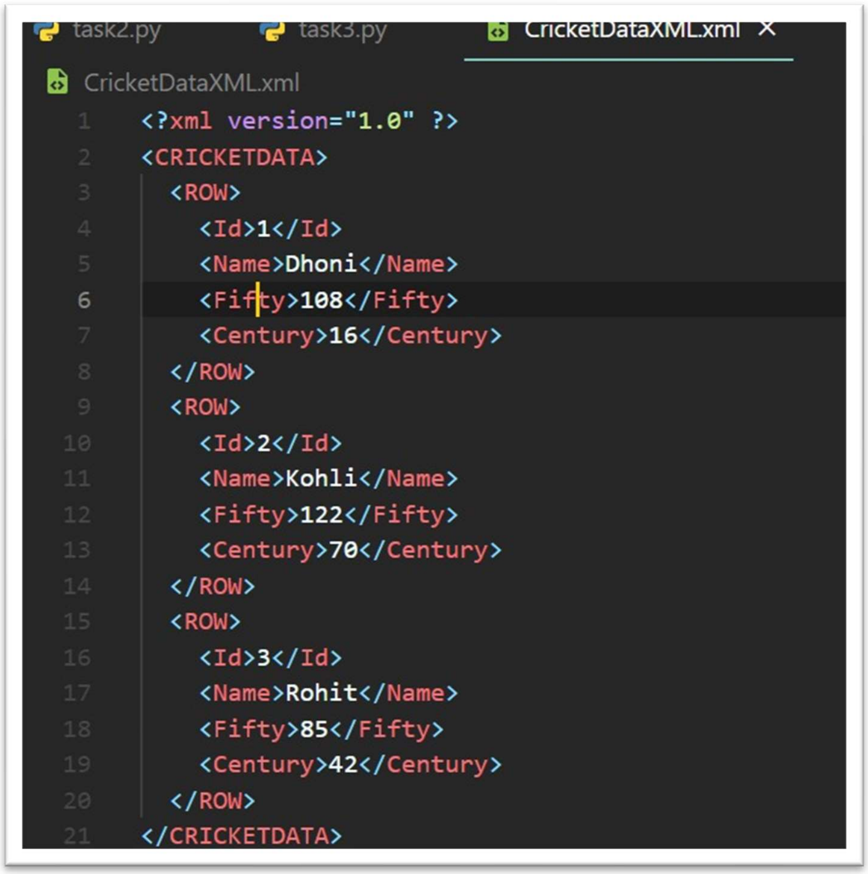
❖ **Code:**


task2.py task3.py × CricketDataXML.xml

task3.py > ...

```python
from msilib.schema import File
import mysql.connector as msql
from lxml.builder import E

conn = msql.connect(host='localhost', user='root',
                    database='BDA_LAB_1', password='')
cursor = conn.cursor()
cursor.execute("SELECT * FROM cricket_data")

outfile = open("CricketDataXML.xml", "w")
rows = cursor.fetchall()
outfile.write('<?xml version="1.0" ?>\n')
outfile.write("<CRICKETDATA>\n")
for row in rows:
    outfile.write("  <ROW>\n")
    outfile.write('    <Id>%s</Id>\n' % row[0])
    outfile.write('    <Name>%s</Name>\n' % row[1])
    outfile.write('    <Fifty>%s</Fifty>\n' % row[2])
    outfile.write('    <Century>%s</Century>\n' % row[3])
    outfile.write('  </ROW>\n')
outfile.write('</CRICKETDATA>\n')
outfile.close()
```

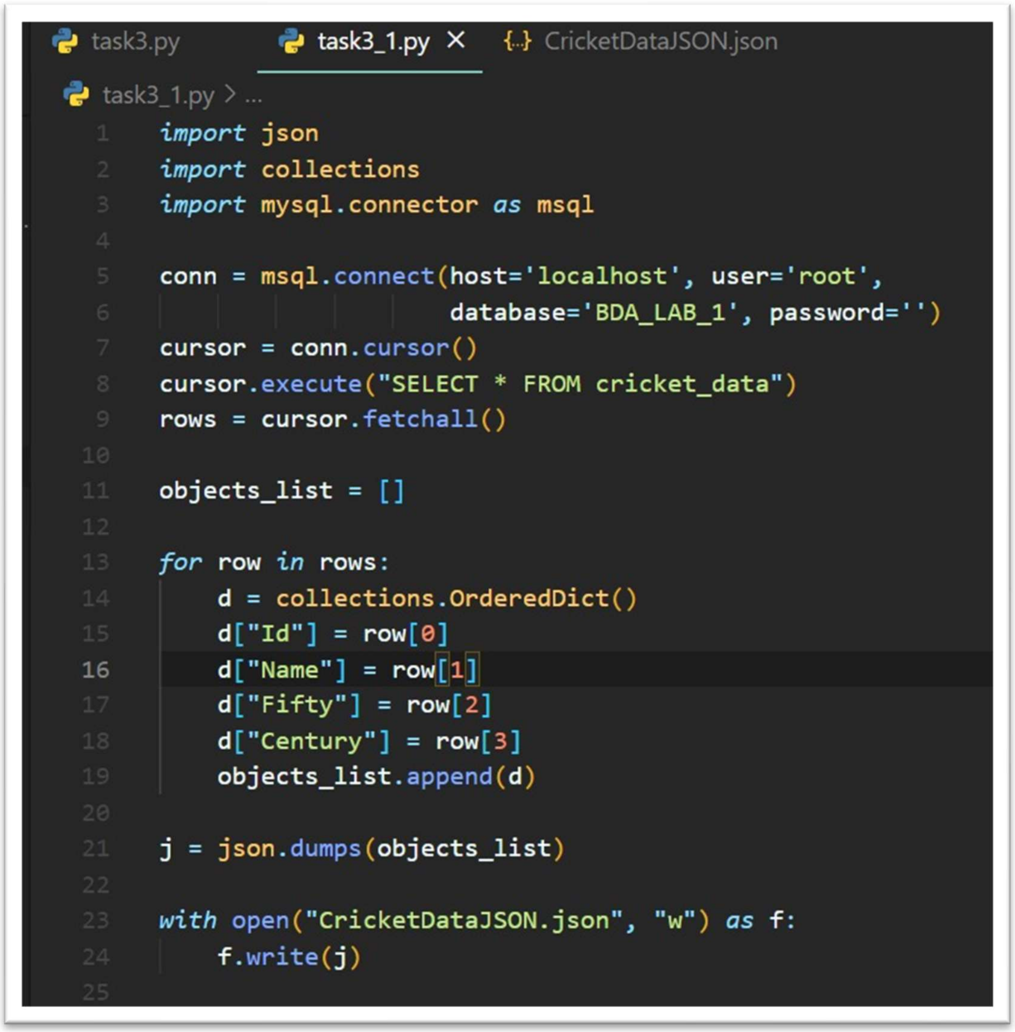
❖ **Output:**



```xml
<?xml version="1.0" ?>
<CRICKETDATA>
  <ROW>
    <Id>1</Id>
    <Name>Dhoni</Name>
    <Fifty>108</Fifty>
    <Century>16</Century>
  </ROW>
  <ROW>
    <Id>2</Id>
    <Name>Kohli</Name>
    <Fifty>122</Fifty>
    <Century>70</Century>
  </ROW>
  <ROW>
    <Id>3</Id>
    <Name>Rohit</Name>
    <Fifty>85</Fifty>
    <Century>42</Century>
  </ROW>
</CRICKETDATA>
```

II. JSON

❖ **Code:**

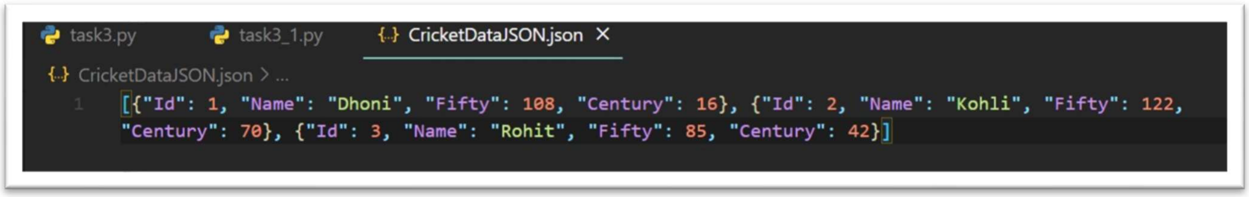


task3.py task3_1.py CricketDataJSON.json

task3_1.py > ...

```python
import json
import collections
import mysql.connector as msql

conn = msql.connect(host='localhost', user='root',
                    database='BDA_LAB_1', password='')
cursor = conn.cursor()
cursor.execute("SELECT * FROM cricket_data")
rows = cursor.fetchall()

objects_list = []

for row in rows:
    d = collections.OrderedDict()
    d["Id"] = row[0]
    d["Name"] = row[1]
    d["Fifty"] = row[2]
    d["Century"] = row[3]
    objects_list.append(d)

j = json.dumps(objects_list)

with open("CricketDataJSON.json", "w") as f:
    f.write(j)
```

❖ **Output:**



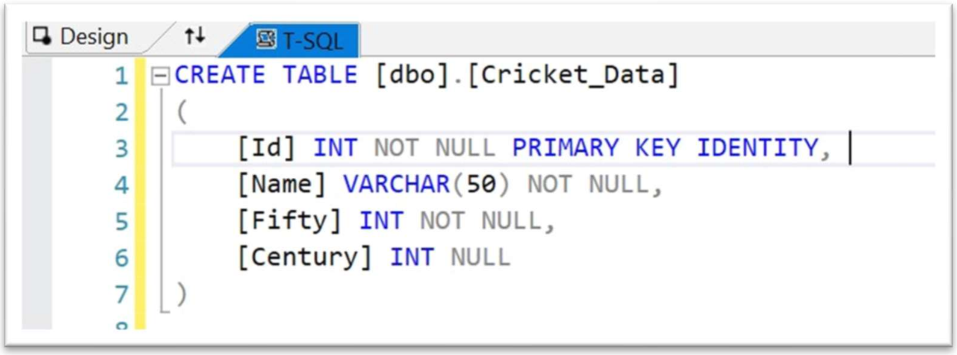
task3.py task3_1.py CricketDataJSON.json

CricketDataJSON.json > ...

```
[{"Id": 1, "Name": "Dhoni", "Fifty": 108, "Century": 16}, {"Id": 2, "Name": "Kohli", "Fifty": 122,
"Century": 70}, {"Id": 3, "Name": "Rohit", "Fifty": 85, "Century": 42}]
```

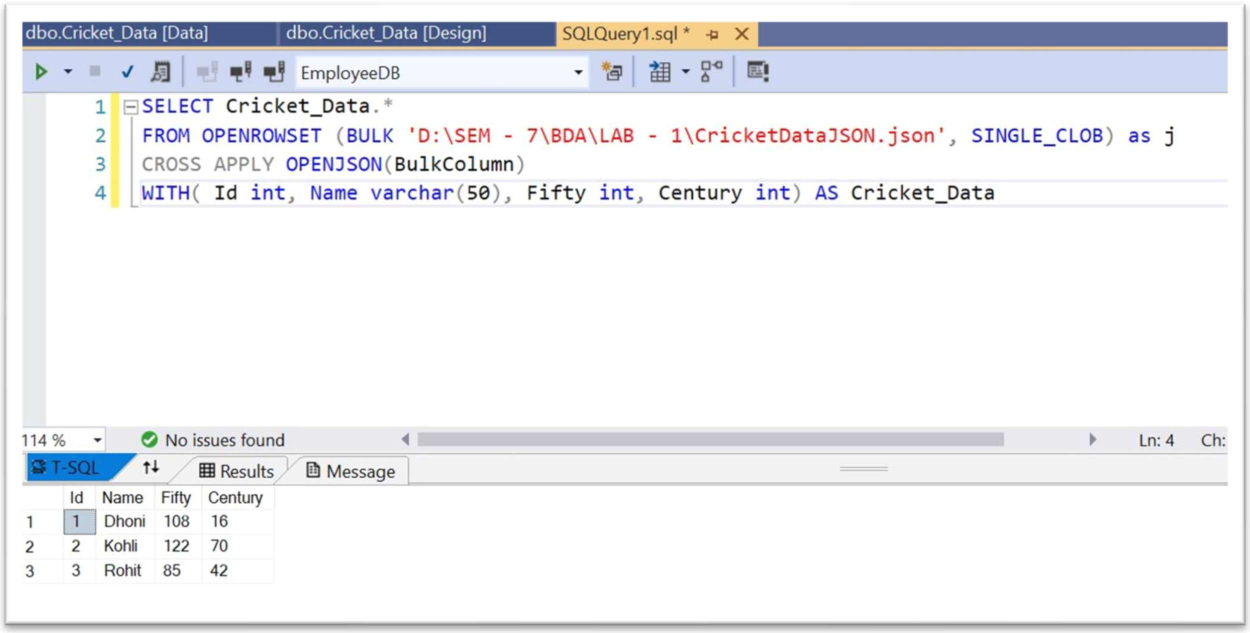
Q. 4: Import XML/JSON file into another database/table.

- Creating table in SQL Server.



```
CREATE TABLE [dbo].[Cricket_Data]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name] VARCHAR(50) NOT NULL,
    [Fifty] INT NOT NULL,
    [Century] INT NULL
)
```

- Query to convert csv to database table.

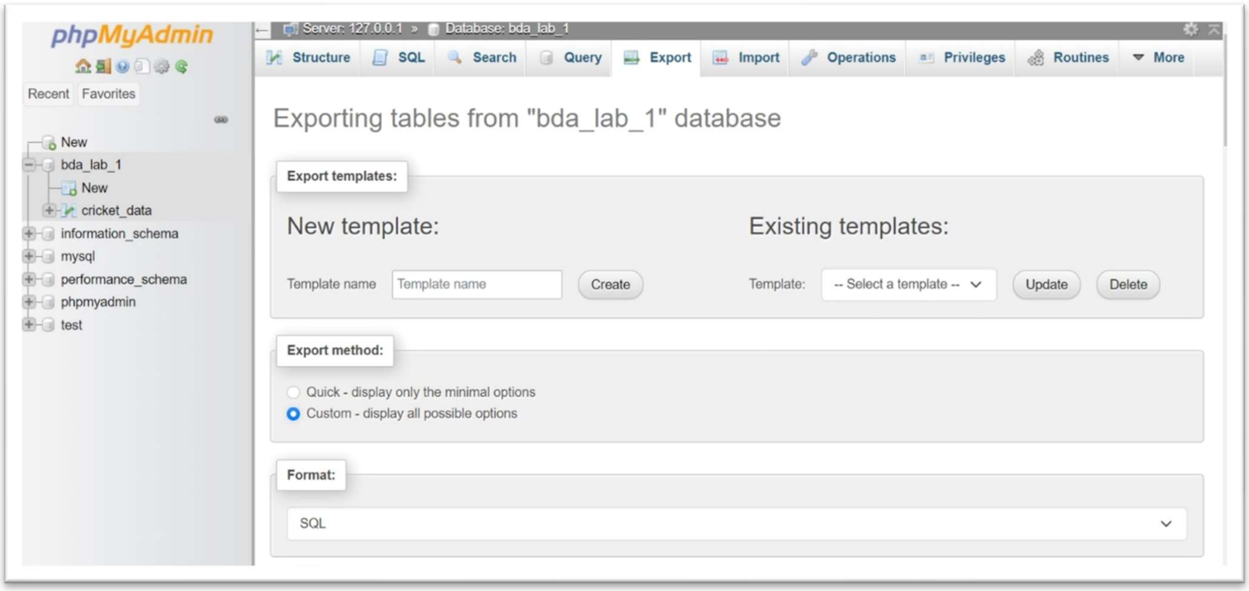


```
SELECT Cricket_Data.*
FROM OPENROWSET (BULK 'D:\SEM - 7\BDA\LAB - 1\CricketDataJSON.json', SINGLE_CLOB) as j
CROSS APPLY OPENJSON(BulkColumn)
WITH( Id int, Name varchar(50), Fifty int, Century int) AS Cricket_Data
```

	Id	Name	Fifty	Century
1	1	Dhoni	108	16
2	2	Kohli	122	70
3	3	Rohit	85	42

Q. 5: Export database dump for data migration/archival.

- ❖ We can easily export any SQL for of database using phpMyAdmin window of Xampp.



Exporting tables from "bda_lab_1" database

Export templates:

New template:

Template name: Template name [Create]

Existing templates:

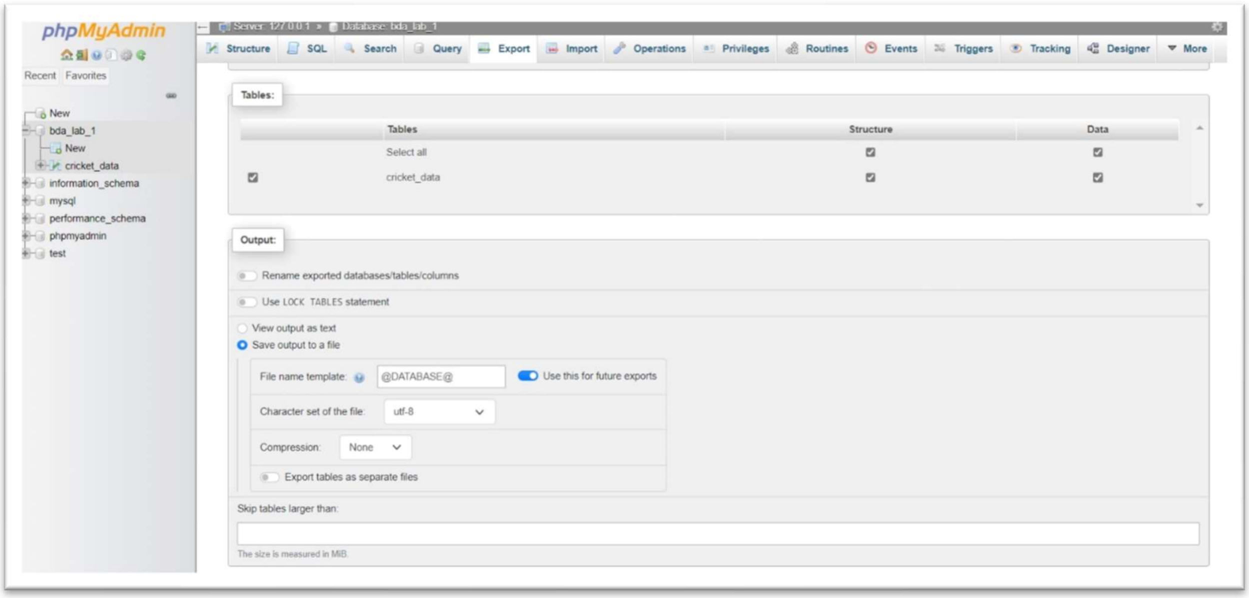
Template: -- Select a template -- [Update] [Delete]

Export method:

- Quick - display only the minimal options
- [✓ Custom - display all possible options]

Format:

SQL



Tables:

	Tables	Structure	Data
	Select all	✓	✓
✓	cricket_data	✓	✓

Output:

- Rename exported databases/tables/columns
- Use LOCK TABLES statement
- View output as text
- [✓ Save output to a file]

File name template: @DATABASE@ — Use this for future exports [✓]

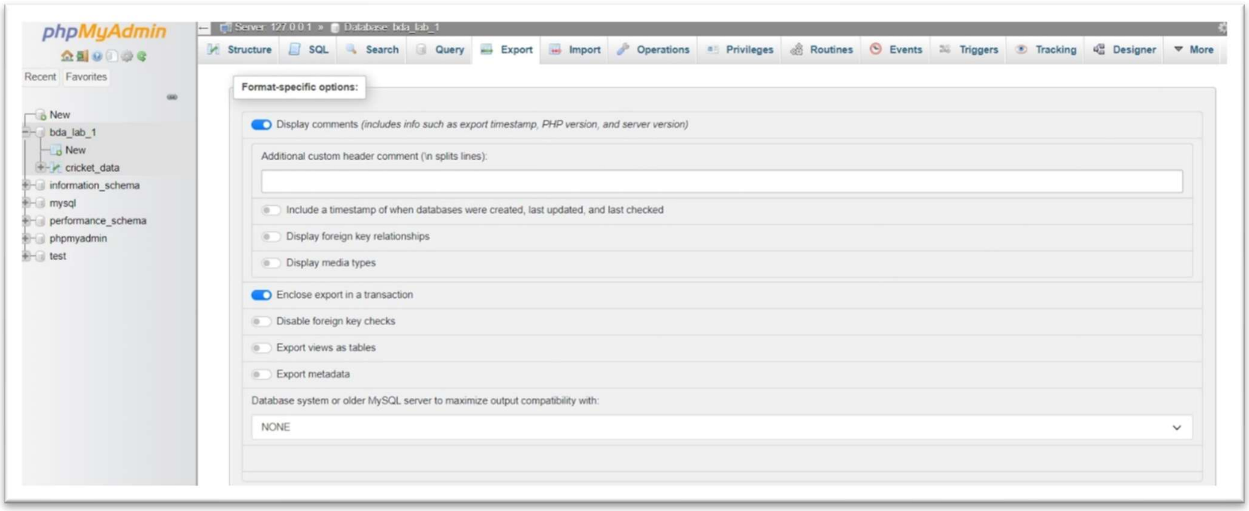
Character set of the file: utf-8

Compression: None

Export tables as separate files

Skip tables larger than:

The size is measured in MiB.


Format-specific options:

[✓] Display comments *(includes info such as export timestamp, PHP version, and server version)*

Additional custom header comment (\n splits lines):

[ ] Include a timestamp of when databases were created, last updated, and last checked

[ ] Display foreign key relationships

[ ] Display media types

[✓] Enclose export in a transaction

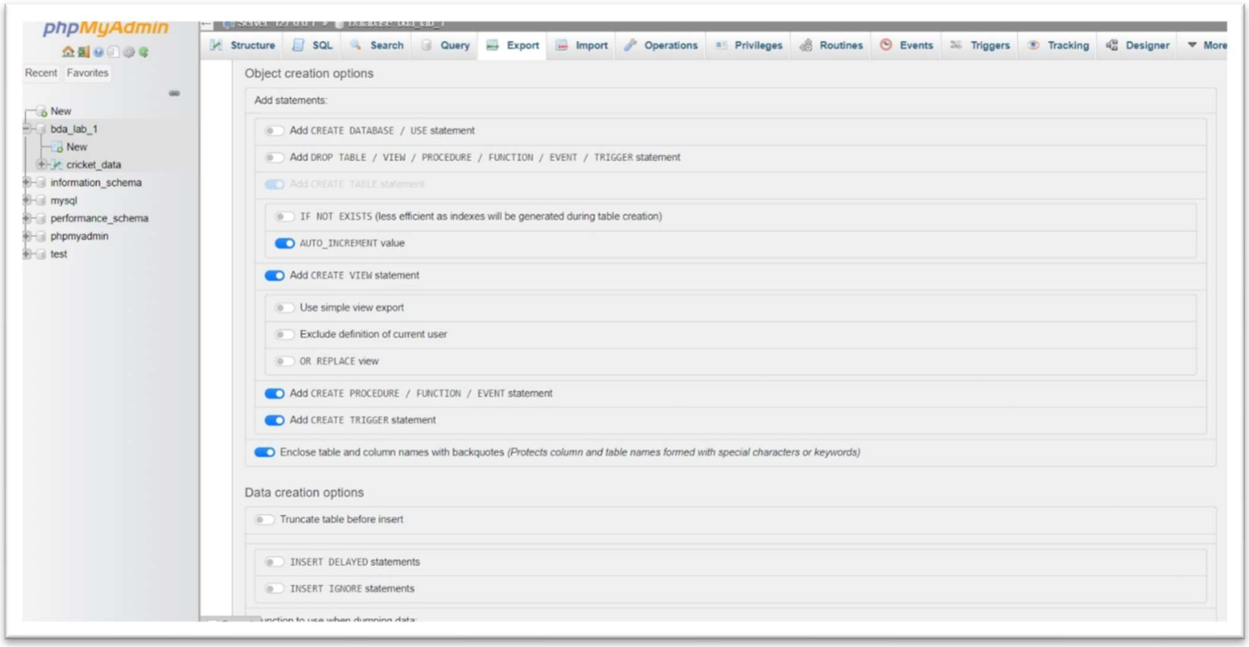
[ ] Disable foreign key checks

[ ] Export views as tables

[ ] Export metadata

Database system or older MySQL server to maximize output compatibility with:

NONE


Object creation options

Add statements:

[ ] Add `CREATE DATABASE / USE` statement

[ ] Add `DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT / TRIGGER` statement

[✓] Add `CREATE TABLE` statement

[ ] `IF NOT EXISTS` (less efficient as indexes will be generated during table creation)

[✓] `AUTO_INCREMENT` value

[✓] Add `CREATE VIEW` statement

[ ] Use simple view export

[ ] Exclude definition of current user

[ ] `OR REPLACE` view

[✓] Add `CREATE PROCEDURE / FUNCTION / EVENT` statement

[✓] Add `CREATE TRIGGER` statement

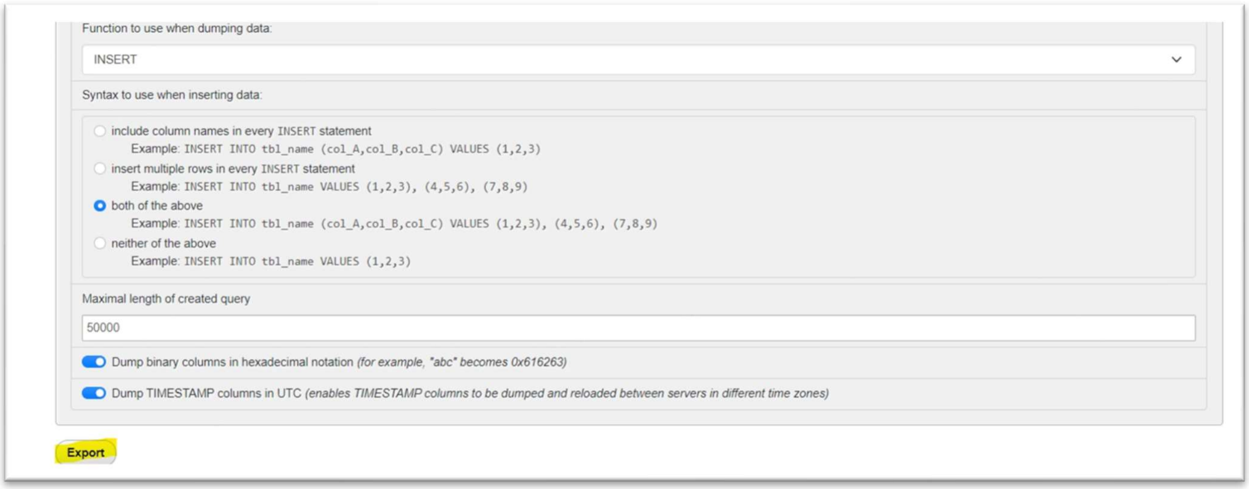
[✓] Enclose table and column names with backquotes *(Protects column and table names formed with special characters or keywords)*

Data creation options

[ ] Truncate table before insert

[ ] `INSERT DELAYED` statements

[ ] `INSERT IGNORE` statements



Function to use when dumping data:

INSERT

Syntax to use when inserting data:

( ) include column names in every INSERT statement
Example: `INSERT INTO tbl_name (col_A,col_B,col_C) VALUES (1,2,3)`
( ) insert multiple rows in every INSERT statement
Example: `INSERT INTO tbl_name VALUES (1,2,3), (4,5,6), (7,8,9)`
[✓ both of the above]
Example: `INSERT INTO tbl_name (col_A,col_B,col_C) VALUES (1,2,3), (4,5,6), (7,8,9)`
( ) neither of the above
Example: `INSERT INTO tbl_name VALUES (1,2,3)`

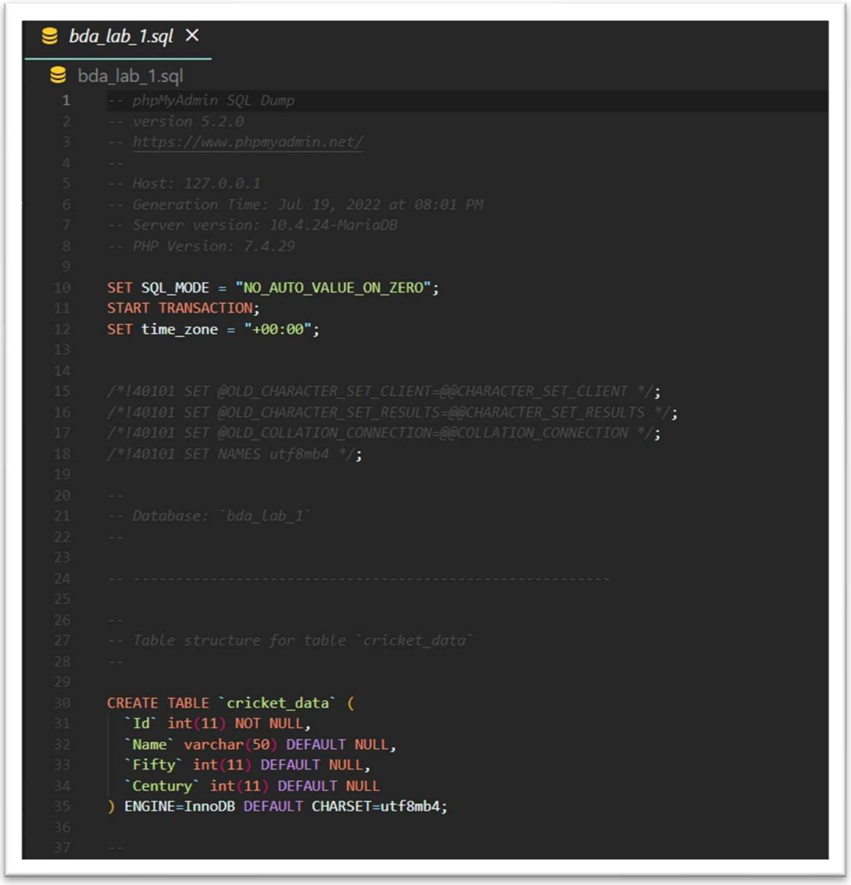
Maximal length of created query

50000

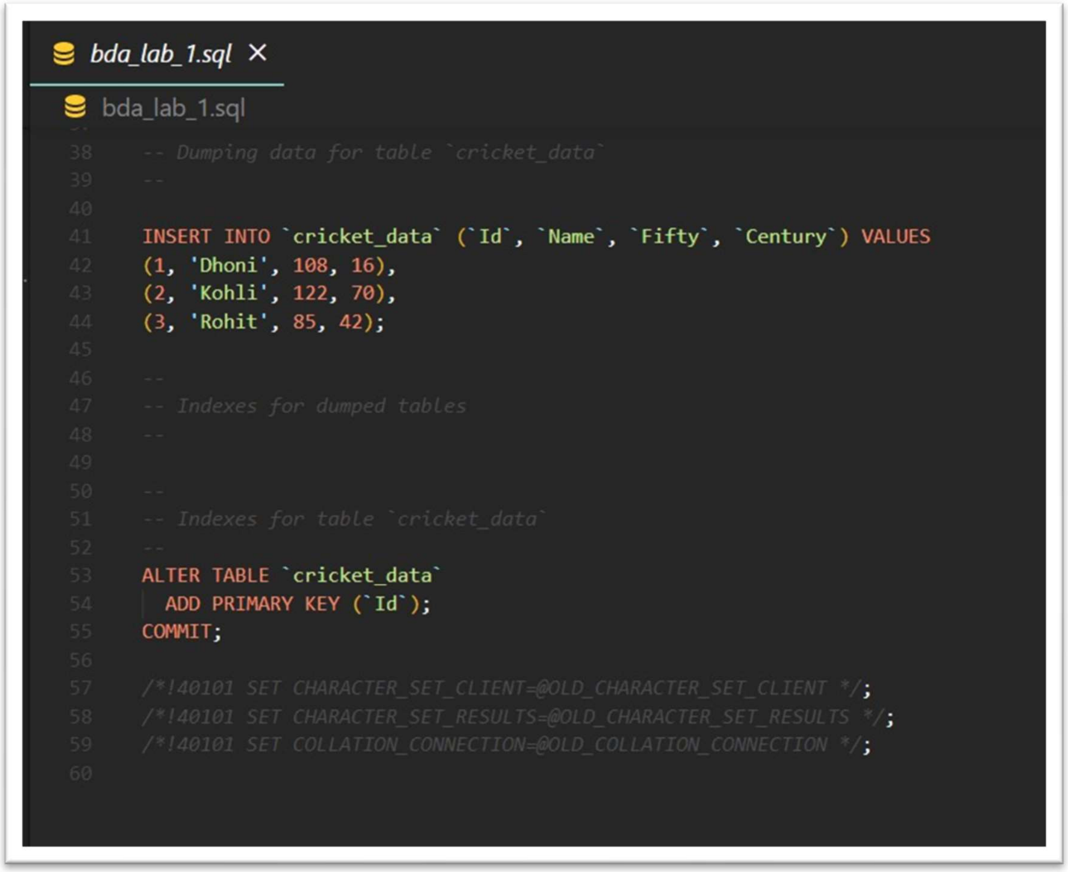
[✓] Dump binary columns in hexadecimal notation *(for example, "abc" becomes 0x616263)*

[✓] Dump TIMESTAMP columns in UTC *(enables TIMESTAMP columns to be dumped and reloaded between servers in different time zones)*

❖ Output file:

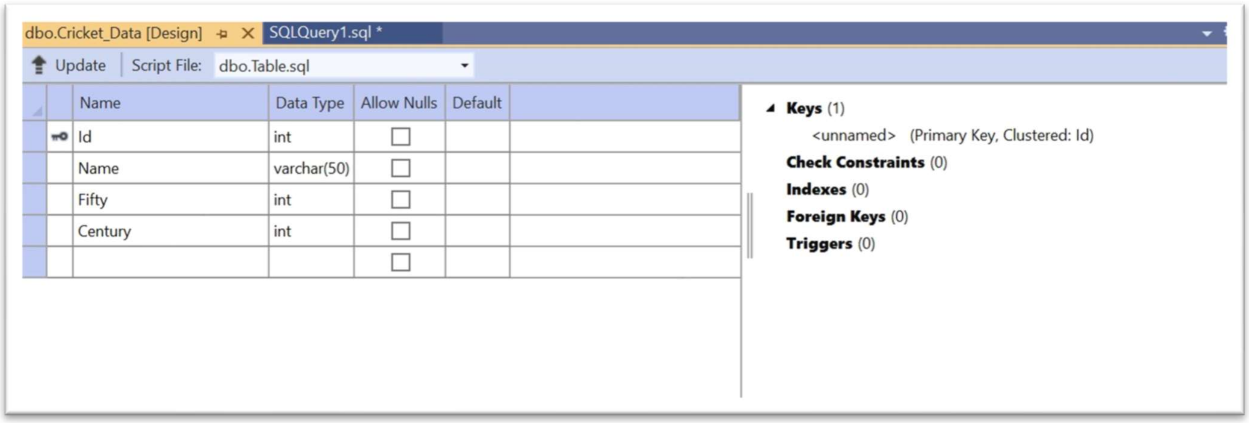


```
-- phpMyAdmin SQL Dump
-- version 5.2.0
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Jul 19, 2022 at 08:01 PM
-- Server version: 10.4.24-MariaDB
-- PHP Version: 7.4.29

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `bda_lab_1`
--

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

--
-- Table structure for table `cricket_data`
--

CREATE TABLE `cricket_data` (
  `Id` int(11) NOT NULL,
  `Name` varchar(50) DEFAULT NULL,
  `Fifty` int(11) DEFAULT NULL,
  `Century` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
```



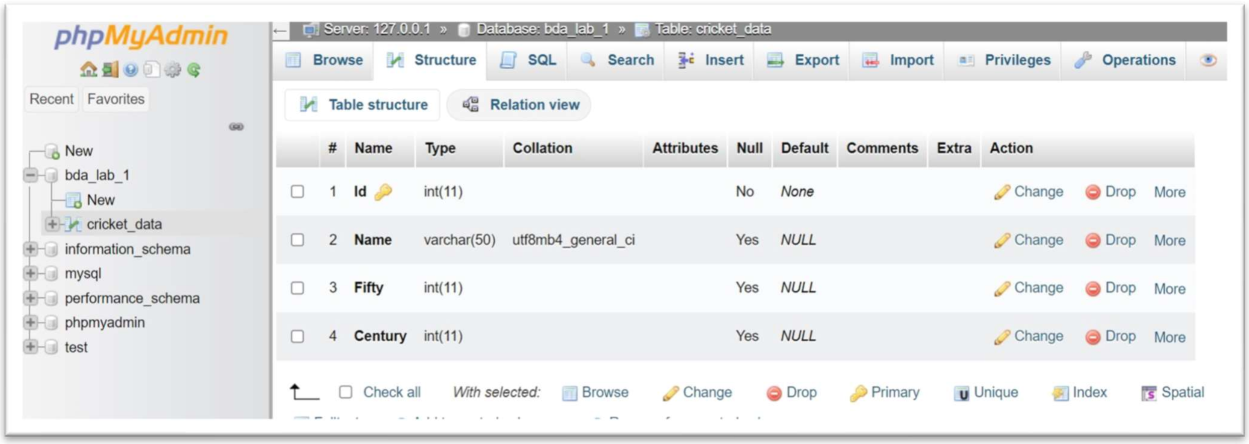
```sql
-- Dumping data for table `cricket_data`
--

INSERT INTO `cricket_data` (`Id`, `Name`, `Fifty`, `Century`) VALUES
(1, 'Dhoni', 108, 16),
(2, 'Kohli', 122, 70),
(3, 'Rohit', 85, 42);

--
-- Indexes for dumped tables
--

--
-- Indexes for table `cricket_data`
--
ALTER TABLE `cricket_data`
  ADD PRIMARY KEY (`Id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Q. 6: Validate/Map data types across different database systems when migrating from one to another.



Name	Data Type	Allow Nulls	Default
Id	int	☐	
Name	varchar(50)	☐	
Fifty	int	☐	
Century	int	☐	
		☐	

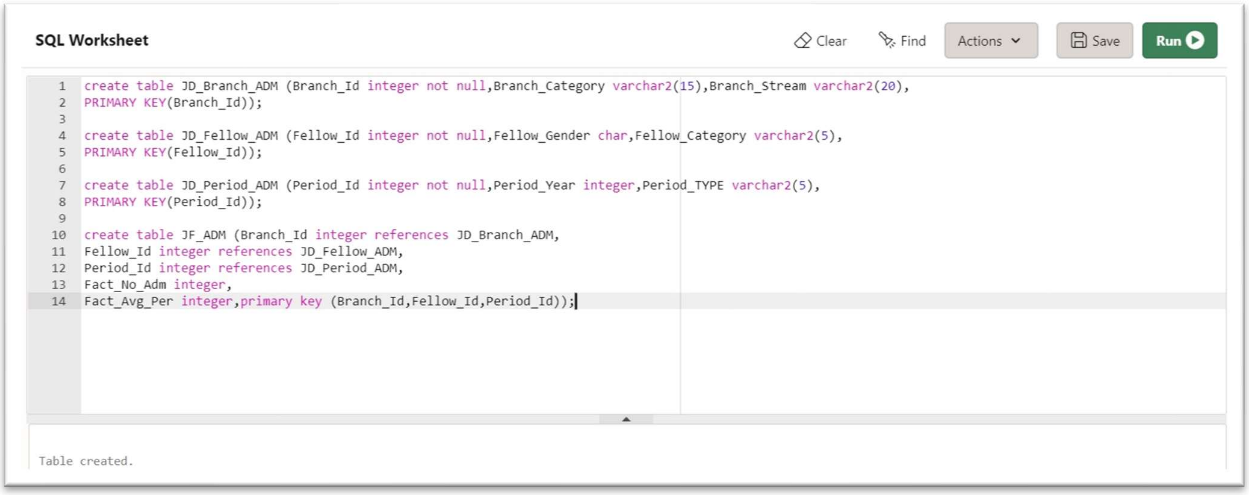
Keys (1)
<unnamed> (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Id	int(11)			No	None			Change Drop More
2	Name	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	Fifty	int(11)			Yes	NULL			Change Drop More
4	Century	int(11)			Yes	NULL			Change Drop More

Q. 7: Represent data cube and perform operations. OLTP – Data Warehouse.

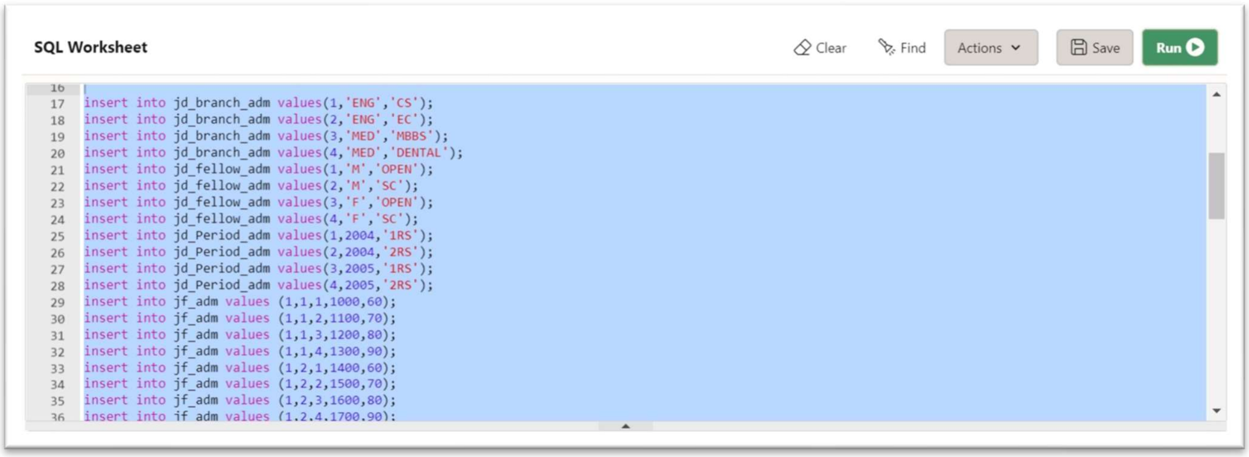
❖ Creating tables:



```
create table JD_Branch_ADM (Branch_Id integer not null,Branch_Category varchar2(15),Branch_Stream varchar2(20),
PRIMARY KEY(Branch_Id));

create table JD_Fellow_ADM (Fellow_Id integer not null,Fellow_Gender char,Fellow_Category varchar2(5),
PRIMARY KEY(Fellow_Id));

create table JD_Period_ADM (Period_Id integer not null,Period_Year integer,Period_TYPE varchar2(5),
PRIMARY KEY(Period_Id));

create table JF_ADM (Branch_Id integer references JD_Branch_ADM,
Fellow_Id integer references JD_Fellow_ADM,
Period_Id integer references JD_Period_ADM,
Fact_No_Adm integer,
Fact_Avg_Per integer,primary key (Branch_Id,Fellow_Id,Period_Id));
```

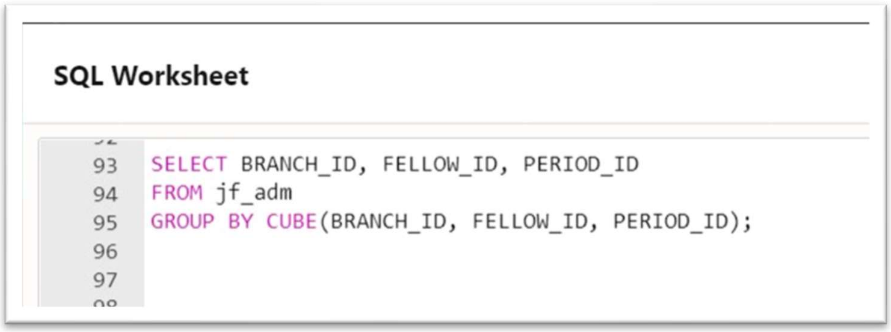
```
Table created.
```

❖ Inserting data into tables:

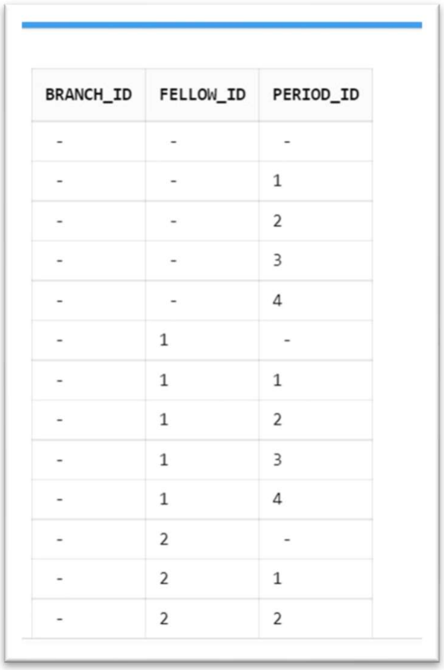


```sql

insert into jd_branch_adm values(1,'ENG','CS');
insert into jd_branch_adm values(2,'ENG','EC');
insert into jd_branch_adm values(3,'MED','MBBS');
insert into jd_branch_adm values(4,'MED','DENTAL');
insert into jd_fellow_adm values(1,'M','OPEN');
insert into jd_fellow_adm values(2,'M','SC');
insert into jd_fellow_adm values(3,'F','OPEN');
insert into jd_fellow_adm values(4,'F','SC');
insert into jd_Period_adm values(1,2004,'1RS');
insert into jd_Period_adm values(2,2004,'2RS');
insert into jd_Period_adm values(3,2005,'1RS');
insert into jd_Period_adm values(4,2005,'2RS');
insert into jf_adm values (1,1,1,1000,60);
insert into jf_adm values (1,1,2,1100,70);
insert into jf_adm values (1,1,3,1200,80);
insert into jf_adm values (1,1,4,1300,90);
insert into jf_adm values (1,2,1,1400,60);
insert into jf_adm values (1,2,2,1500,70);
insert into jf_adm values (1,2,3,1600,80);
insert into jf_adm values (1,2,4,1700,90);
```

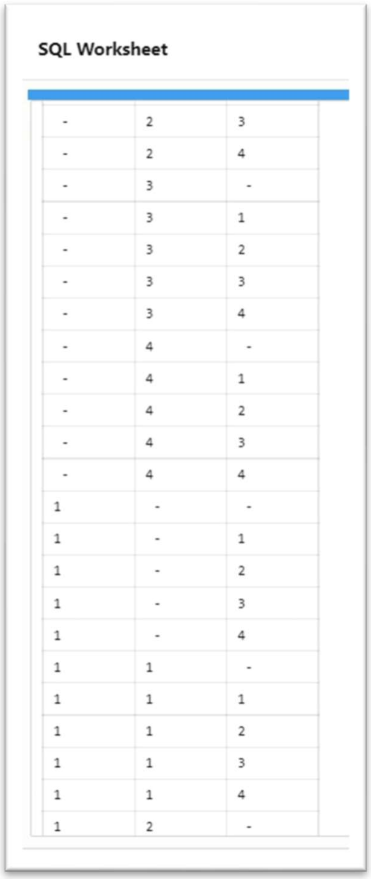
❖ Representing data as data cube:



```sql
SELECT BRANCH_ID, FELLOW_ID, PERIOD_ID
FROM jf_adm
GROUP BY CUBE(BRANCH_ID, FELLOW_ID, PERIOD_ID);
```

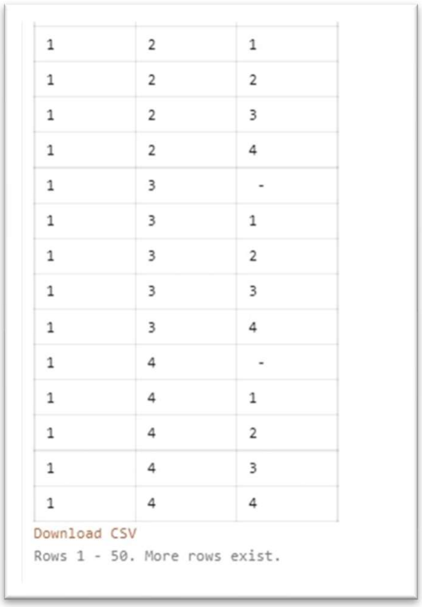


BRANCH_ID	FELLOW_ID	PERIOD_ID
-	-	-
-	-	1
-	-	2
-	-	3
-	-	4
-	1	-
-	1	1
-	1	2
-	1	3
-	1	4
-	2	-
-	2	1
-	2	2



SQL Worksheet

-	2	3
-	2	4
-	3	-
-	3	1
-	3	2
-	3	3
-	3	4
-	4	-
-	4	1
-	4	2
-	4	3
-	4	4
1	-	-
1	-	1
1	-	2
1	-	3
1	-	4
1	1	-
1	1	1
1	1	2
1	1	3
1	1	4
1	2	-

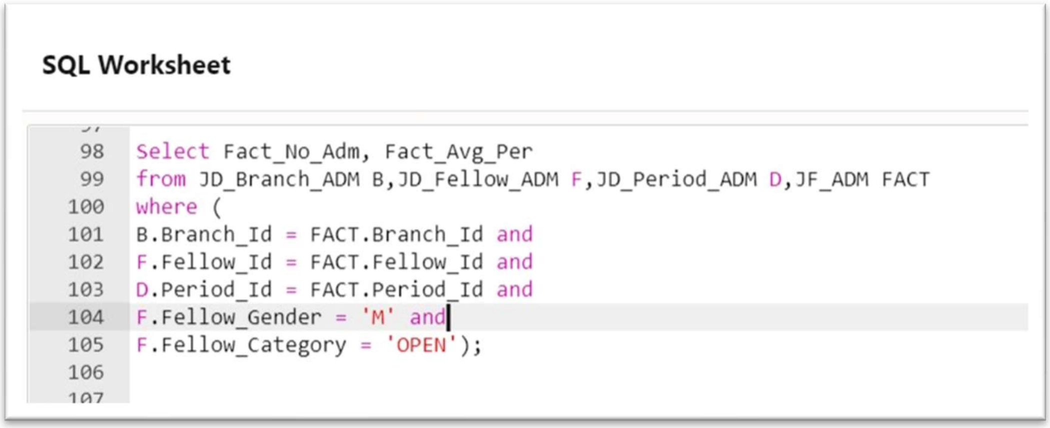


1	2	1
1	2	2
1	2	3
1	2	4
1	3	-
1	3	1
1	3	2
1	3	3
1	3	4
1	4	-
1	4	1
1	4	2
1	4	3
1	4	4

Download CSV

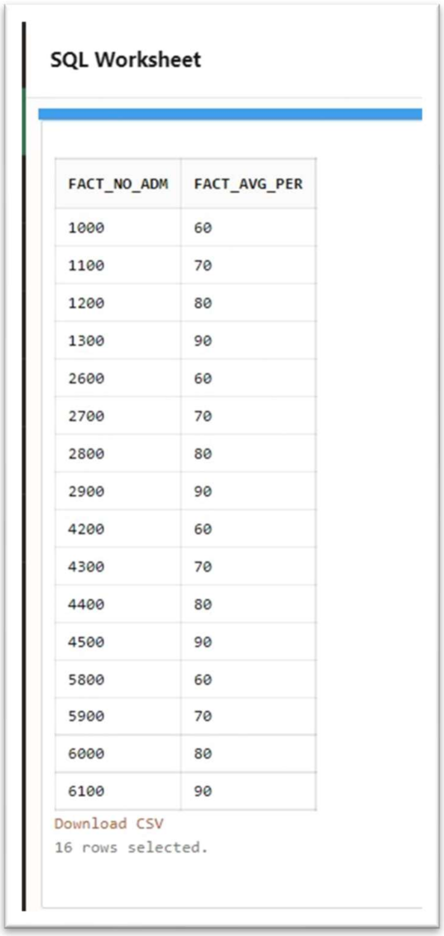
Rows 1 - 50. More rows exist.

❖ Slice operation:



SQL Worksheet

```
Select Fact_No_Adm, Fact_Avg_Per
from JD_Branch_ADM B,JD_Fellow_ADM F,JD_Period_ADM D,JF_ADM FACT
where (
B.Branch_Id = FACT.Branch_Id and
F.Fellow_Id = FACT.Fellow_Id and
D.Period_Id = FACT.Period_Id and
F.Fellow_Gender = 'M' and
F.Fellow_Category = 'OPEN');
```



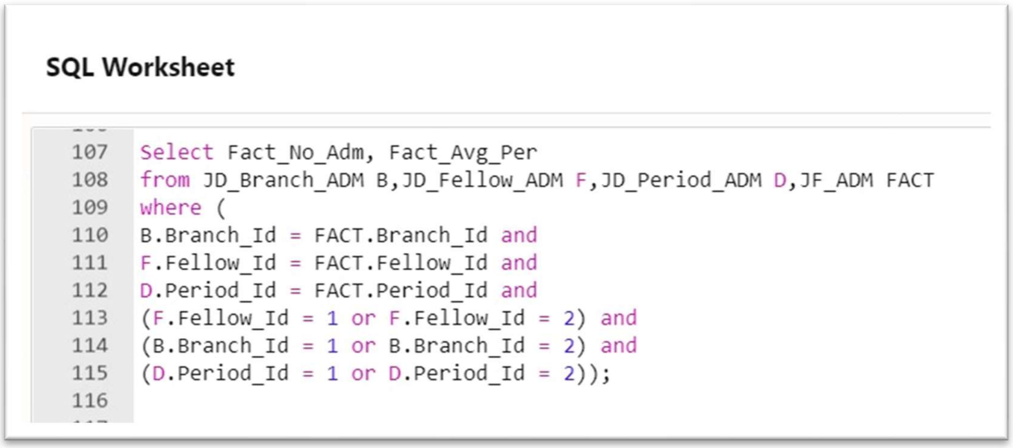
SQL Worksheet

FACT_NO_ADM	FACT_AVG_PER
1000	60
1100	70
1200	80
1300	90
2600	60
2700	70
2800	80
2900	90
4200	60
4300	70
4400	80
4500	90
5800	60
5900	70
6000	80
6100	90

Download CSV

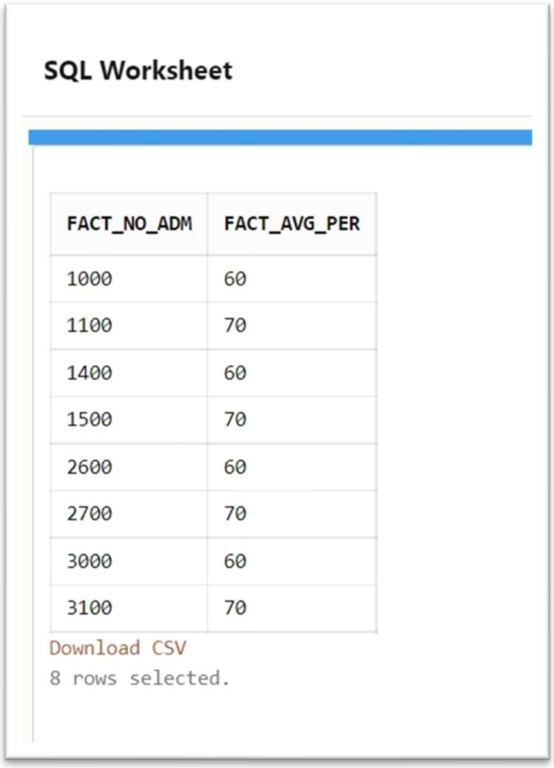
16 rows selected.

❖ **Dice operation:**



SQL Worksheet

```
Select Fact_No_Adm, Fact_Avg_Per
from JD_Branch_ADM B,JD_Fellow_ADM F,JD_Period_ADM D,JF_ADM FACT
where (
B.Branch_Id = FACT.Branch_Id and
F.Fellow_Id = FACT.Fellow_Id and
D.Period_Id = FACT.Period_Id and
(F.Fellow_Id = 1 or F.Fellow_Id = 2) and
(B.Branch_Id = 1 or B.Branch_Id = 2) and
(D.Period_Id = 1 or D.Period_Id = 2));
```

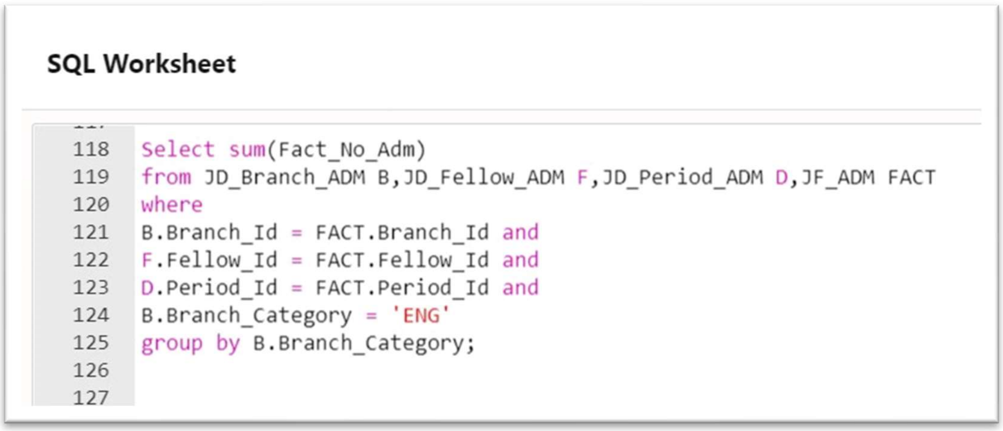

SQL Worksheet

FACT_NO_ADM	FACT_AVG_PER
1000	60
1100	70
1400	60
1500	70
2600	60
2700	70
3000	60
3100	70

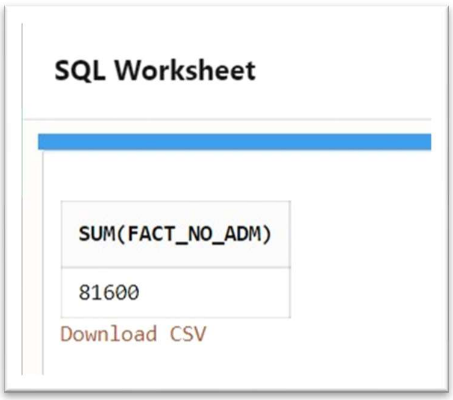
Download CSV

8 rows selected.

❖ **Roll up operation:**


SQL Worksheet

```
Select sum(Fact_No_Adm)
from JD_Branch_ADM B,JD_Fellow_ADM F,JD_Period_ADM D,JF_ADM FACT
where
B.Branch_Id = FACT.Branch_Id and
F.Fellow_Id = FACT.Fellow_Id and
D.Period_Id = FACT.Period_Id and
B.Branch_Category = 'ENG'
group by B.Branch_Category;
```

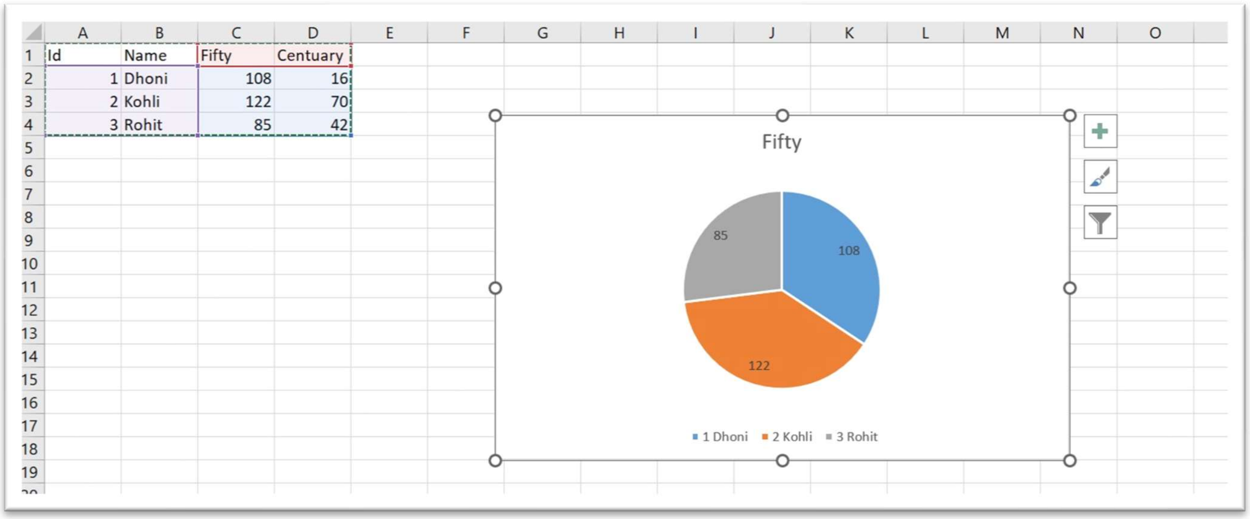


SQL Worksheet

SUM(FACT_NO_ADM)
81600

Download CSV

Q. 8: Generate pdf report. / Use any virtualization tool.



Id	Name	Fifty	Centuary
1	Dhoni	108	16
2	Kohli	122	70
3	Rohit	85	42