MAY

2024

ON-JOB TRAINING REPORT

Training in Data Analytics

NFB
National Facility for Biopharmaceuticals
Transforming Education into Expertise

Prepared By:

Prarthi Kothari

M. Sc. Bioinformatics - Part I

Roll No. - 115

SGCP's

Guru Nanak Khalsa College

of Arts, Science & Commerce (Autonomous)

ACKNOWLEDGEMENT

I would like to convey my profound appreciation to the exceptional group of instructors at the National Facility for Biopharmaceuticals (NFB) for all of their assistance and encouragement during our practical training program.

I sincerely thank Ms. Valencia D'Souza, Senior Research Fellow and Ms. Ira Kode, the student coordinator for the program. Your tireless efforts, innovative ideas, and unwavering commitment have made this course a resounding success. We appreciate the countless hours spent planning, developing, and executing the curriculum, ensuring an enriching learning experience for all participants. Your dedication to fostering a collaborative and supportive environment has been truly inspiring. We extend our heartfelt thanks for your invaluable contributions to the success of this course.

I would like to express my sincere gratitude for the invaluable contributions of Mr. Mandar Katkar, a Data Science Trainer at DeveLearn, who taught me the fundamental concepts of Python programming language and SQL. His engaging lectures and hands-on exercises helped me grasp the core principles and practical applications of these essential tools for data analysis and manipulation. I am grateful for his expertise, patience, and commitment to empowering me with the skills needed to thrive in the field of data science.

Mrs. Sushma Ghadge, an MIS and Graphics Teacher at DeveLearn, imparted valuable knowledge and skills that empowered us to effectively utilize MS-Excel, a spreadsheet software and design visual aids. Her patient guidance and practical approach enabled us to master the fundamentals and apply them confidently in our academic and professional pursuits.

I learned how Power Query and Power BI work from Ms. Sanvee Khot, a Data Science Trainer at DeveLearn. My ability to effectively extract, manipulate, and analyze complicated data sets was made possible by her proficiency with these potent data analysis tools. My comprehension of utilizing Power Query and Power BI to extract meaningful insights was much enhanced by Ms. Sanvee's interactive teaching style and practical methodology. I appreciate all of her help and dedication in providing me with the knowledge and abilities required to succeed in the data science industry.

I express my gratitude to Dr. Gursimran Kaur Uppal, who oversees the Bioinformatics Department at Guru Nanak Khalsa College of Arts, Science, and Commerce, for providing me with unwavering support and motivation. Their guidance has been really helpful in helping me get beyond the obstacles I've had in my academic career.

I would like to express my gratitude to my teachers, Mrs. Aparna Patil Kose and Mrs. Sermarani Nadar, for their insightful advice and unwavering support during my master's program. My educational experience has been greatly enhanced by their knowledge and assistance. Finally, I would want to sincerely thank my friends, family, and coworkers for their steadfast understanding and support during my studies. Their inspiration and support have helped me stay motivated and focused on achieving my objectives.

INDEX

Chapter No.	Chapter Title	Page No.
1	Schedule And Training Details	1
2	Data Analytics	4
3	Python Programming Language	7
4	SQL (Structured Query Language)	15
5	Advanced Excel	37
6	Power Bi	42
7	Skills Acquired	47
8	Conclusion	48
9	References	49

CHAPTER 1 – SCHEDULE AND TRAINING DETAILS

Part 1: Introduction to Python Programming Language

Date	Day	Session Information	Details	Mentor
02-05-2024	Thursday	**Day 1:** Basics of Python Programming Language	Data Types, Variables and their Nomenclature, String and String Handling, Comments	Mr. Mandar Katkar (Data Science Trainer)
03-05-2024	Friday	**Day 2:** Basics of Python Programming Language	User Input, Type Casting (Implicit and Explicit Type Conversion), Operators (Arithmetic, Relational, Logical and Membership Operators), Lists, Dictionary, Set, Tuple	
04-05-2024	Saturday	**Day 3:** Basics of Python Programming Language	Conditional Statements (if, if-else, if-elif-else, Nested if), Iterative Statements / Loops (for loop)	
06-05-2024	Monday	**Day 4:** Basics of Python Programming Language	Iterative Statements / Loops (while loop), Functions	
07-05-2024	Tuesday	**Day 5:** Analysis and Manipulation of Data using Python Libraries	Data Structures, Data Handling and Manipulation of Data Frames using Pandas	
08-05-2024	Wednesday	**Day 6:** Analysis and Manipulation of Data using Python Libraries	Creating and Manipulating Matrices using NumPy and Visualizing Data using EDA (Exploratory Data Analysis)	
09-05-2024	Thursday	**Day 7:** Analysis and Manipulation of Data using Python Libraries	Visualizing Data using EDA (Exploratory Data Analysis) and Matplotlib	

Part 2: Introduction to SQL (Structured Query Language)

Date	Day	Session Information	Details	Mentor
10-05-2024	Friday	**Day 8:** SQL - Data Manipulation and Management	Introduction to SQL (Structured Query Language), Creating and Dropping a Database and a Table	Mr. Mandar Katkar (Data Science Trainer)
11-05-2024	Saturday	**Day 9:** SQL - Data Manipulation and Management	Inserting Values in a Table and Displaying the Table, WHERE Clause and Distinct Clause), Operators (BETWEEN, AND, OR, IN, LIKE)	
13-05-2024	Monday	**Day 10:** SQL - Data Manipulation and Management	GROUP BY, ORDER BY, JOINS (Inner Join, Left Join, Right Join, Full Outer Join)	
14-05-2024	Tuesday	**Day 11:** SQL - Data Manipulation and Management	Aggregate functions (AVERAGE, MAX, MIN, COUNT), HAVING Clause, LIMIT Clause, Subquery	
15-05-2024	Wednesday	**Day 12:** SQL - Data Manipulation and Management	UNION, Intersection, UPDATE, ADD COLUMN, DROP COLUMN, TRUNCATE TABLE, Duplicate Table using AS Clause, ROLLBACK and COMMIT	

Part 3: Introduction to Advanced Excel

Date	Day	Session Information	Details	Mentor
16-05-2024	Thursday	**Day 13:** Advanced Excel	Basics of Excel (Adding numerical values, Filling, Data Splitting)	Mrs. Sushma Ghadge (MIS and Graphics Teacher)
17-05-2024	Friday	**Day 14:** Advanced Excel	Transposing, Sorting and Filtering Data, Drop Down Lists	
18-05-2024	Saturday	**Day 15:** Advanced Excel	Inserting Charts, Pivot Tables	
20-05-2024	Monday	**Day 16:** Advanced Excel	Formulas (Add, Subtract, Multiply, Divide)	
21-05-2024	Tuesday	**Day 17:** Advanced Excel	Functions (SUM, AVERAGE, MIN, MAX)	
22-05-2024	Wednesday	**Day 18:** Advanced Excel	Date and Time, Joining Text and Numbers), IF Statements	
23-05-2024	Thursday	**Day 19:** Advanced Excel	VLOOKUP, Conditional Functions	

Part 4: Introduction to Power BI

Date	Day	Session Information	Details	Mentor
24-05-2024	Friday	**Day 20:** Introduction to Power BI	Basics of Power BI	Ms. Sanvee Khot (Data Science Trainer)
25-05-2024	Saturday	**Day 21:** Introduction to Power BI	Learning about the Dataset and Handling the Dataset	
27-05-2024	Monday	**Day 22:** Introduction to Power BI	Manipulating and Editing Data using Power Query	
28-05-2024	Tuesday	**Day 23:** Introduction to Power BI	Model View of the Dataset	
29-05-2024	Wednesday	**Day 24:** Introduction to Power BI	Calculating Measures using DAX Queries	
30-05-2024	Thursday	**Day 25:** Introduction to Power BI	Visualization of Dataset by designing a Dashboard	
31-05-2024	Friday	**Day 26:** Introduction to Power BI	Review of Power BI Assignment	
03-06-2024	Monday	**Day 27**	Exam	

CHAPTER 2 – DATA ANALYTICS

Data analytics is a crucial field that involves collecting, processing, and analyzing large datasets to uncover meaningful insights and patterns. It plays a significant role in various scientific disciplines by enabling researchers to make data-driven decisions, optimize processes, and drive innovation.

Role of Data Analytics in Scientific Field

Data analytics has transformed scientific research by providing powerful tools and techniques to handle the vast amounts of data generated in modern scientific studies. It enables scientists to:

1. Analyze experimental data to test hypotheses and draw conclusions using statistical analysis
2. Identify patterns and relationships in complex datasets that may lead to new discoveries or theories
3. Develop predictive models to forecast future outcomes or behaviors
4. Optimize experimental designs and procedures for better efficiency and accuracy
5. Collaborate with domain experts to translate data insights into actionable knowledge
6. Communicate findings more effectively through data visualization

Advantages of Data Analytics

1. **Enhanced decision-making:** Data analytics provides objective, evidence-based insights to support scientific decision-making, reducing the reliance on intuition or anecdotal evidence.
2. **Improved efficiency:** By automating data collection, processing, and analysis tasks, data analytics can significantly streamline scientific workflows along with the ability to handle large, complex datasets that would be difficult to analyze manually.
3. **Increased reproducibility:** The use of standardized data analytics techniques and tools can enhance the reproducibility of scientific studies, a crucial aspect of the scientific method.
4. **Interdisciplinary collaboration:** Data analytics serves as a common language that facilitates collaboration among researchers from different fields, enabling the emergence of new research directions by identification of unexpected relationships and insights.

Disadvantages of Data Analytics

1. **Data quality and integrity:** The accuracy and reliability of data analytics results depend on the quality of the input data. Poor data collection methods, inconsistencies, or errors can lead to misleading conclusions. Hence, there is a potential for misinterpretation of results due to biases or errors in the data or analysis.
2. **Ethical considerations:** The use of data analytics in science raises important ethical questions, such as data privacy, informed consent, and the potential for misuse or misinterpretation of findings.
3. **Skill gaps:** Effective data analytics requires a combination of technical skills (e.g., programming, statistics) and domain-specific knowledge. Bridging this skill gap can be challenging, particularly in fields where data analytics is a relatively new approach.
4. **Computational resources:** Analyzing large, complex datasets often requires significant computational power and storage capacity, which can be costly and may not be readily available to all researchers.

Applications of Data Analytics

Data analytics has applications in virtually every scientific discipline, from biology and medicine to astronomy and climate science. Some examples include:

1. **Genomics:** Analyzing DNA sequences to identify genetic variations associated with diseases or traits
2. **Neuroscience:** Studying brain activity patterns to understand cognitive processes and disorders
3. **Particle physics:** Detecting patterns in particle collisions to uncover new fundamental particles or laws of nature
4. **Ecology:** Modeling ecosystem dynamics and predicting the impact of environmental changes
5. **Astronomy:** processing and interpreting data from telescopes and satellites

Future Prospects of Data Analytics

As data generation continues to accelerate and computational power becomes more accessible, the role of data analytics in science is expected to grow even more significant in the coming years. Some emerging trends and future prospects include:

1. **Integration with artificial intelligence and machine learning:** The combination of data analytics with advanced AI techniques will enable automation and more sophisticated data-driven discoveries and predictions, along with integration of data analytics with other emerging technologies like the Internet of Things (IoT) and cloud computing.

2. **Increased emphasis on data management and governance:** The need for robust data management practices and policies will become more critical as the volume and complexity of scientific data continue to increase.

3. **Interdisciplinary collaboration and convergence:** Data analytics will continue to serve as a catalyst for collaboration among researchers from different fields, leading to the emergence of new interdisciplinary areas of study.

4. **Democratization of data analytics tools:** The development of user-friendly, cloud-based data analytics platforms will make these tools more accessible to researchers across disciplines.

In conclusion, data analytics has become an indispensable tool in modern scientific research, enabling researchers to extract meaningful insights from vast amounts of data and drive innovation across disciplines. Data analytics has the potential to accelerate scientific discoveries, improve decision-making, and drive innovation in various fields. While there are challenges and limitations to consider, the future prospects of data analytics in science are bright, with the potential to revolutionize the way we understand and interact with the world around us.

CHAPTER 3 – PYTHON PROGRAMMING LANGUAGE

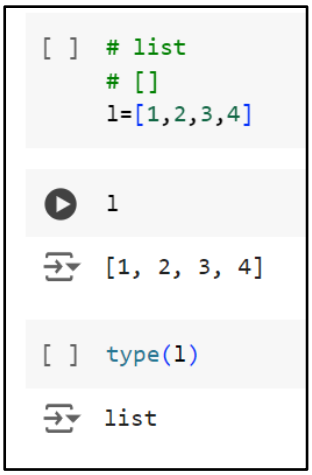
Mentor: Mr. Mandar Katkar (Data Science Trainer)

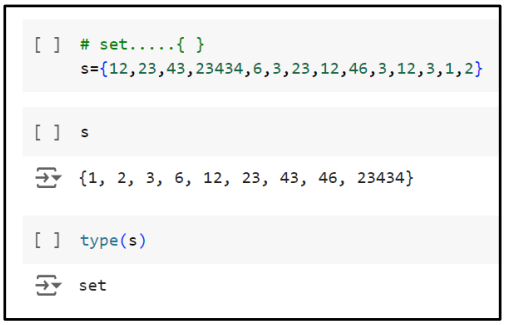
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was developed by Guido van Rossum and first released in 1991.

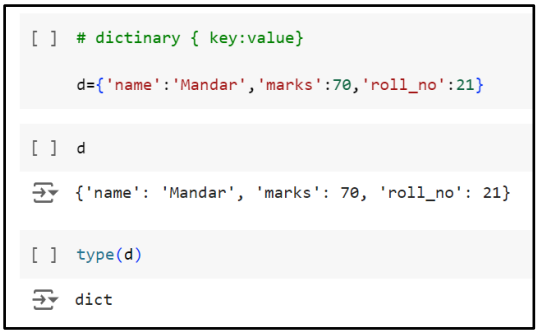
Category	Property	Description
Language Fundamentals	Interpreted	Code runs line by line without prior compilation.
	Object-Oriented	Supports object-based programming concepts like classes and objects.
	High-Level	Abstractions hide machine-level details, making code easier to read and write.
	Dynamic Semantics	Variable types are determined at runtime, offering flexibility.
Code Readability and Flexibility	Emphasizes Readability	Uses indentation to define code blocks, improving clarity.
	Supports Multiple Paradigms	Can be used for structured, object-oriented, and functional programming styles.
Versatile Standard Library	Batteries Included	Provides built-in modules for common tasks, reducing development time.
	Wide Functionality	Offers tools for file handling, data manipulation, networking, and more.
Dynamic Typing and Memory Management	Dynamically Typed	Variables don't require pre-defined types, offering flexibility.
	Automatic Memory Management	Handles memory allocation and deallocation, simplifying development.
Active Community and Ecosystem	Large Third-Party Library Support	Numerous libraries and frameworks extend Python's functionality significantly.
	Python Package Index (PyPI)	Central repository for finding and installing these extensions.
Broad Range of Applications	Server-Side Web Development	Creates dynamic web content.
	Software Development	Used for building various software applications.
	Data Analysis	Powerful tool for data exploration and manipulation.
	Artificial Intelligence	Popular language for machine learning and AI development.
	System Scripting	Automates tasks within computer systems.

Python Data Types

Category	Types	Description	Examples
Numeric Types	int (Integers)	Whole numbers	5, 10, -3
	float (Floats)	Numbers with decimals	2.5, -0.1, 3.14
	complex	Numbers with real and imaginary parts	2 + 3j, -4j
Boolean Type	bool	True or False values	True, False
String Type	str (Strings)	Text data in quotes	"Hello, World!", 'Python is cool'
Sequence Types	list	Ordered collections of items in square brackets	[1, 2, 3], ['mango', 'pineapple']
	tuple	Immutable ordered collections in parentheses	(1, 2, 3), ('red', 'green'))
	range	Sequence of numbers for loops	range(5) for looping 5 times
Mapping Type	dict (Dictionary)	Key-value pairs in curly braces	{1: 'This', 2: 'is', 3: 'a dictionary'}
Set Type	set	Unordered collections of unique items in curly braces	{1, 2, 'apple'}







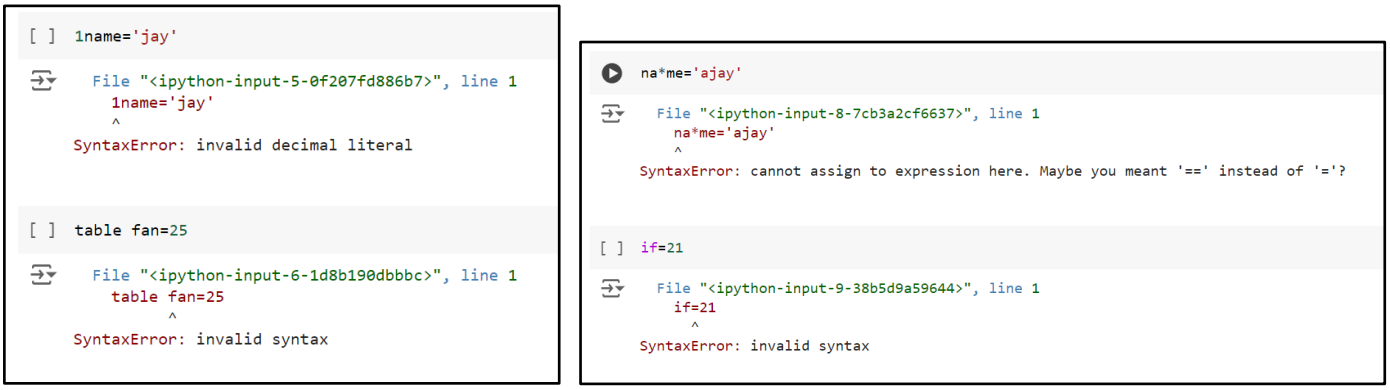
Variables in Python

Variables are used to store data values in Python. To create a variable, you need to give it a name and assign a value to it using the assignment operator =.

Variable Naming Conventions

1. Variable names are case-sensitive, so age, Age, and AGE are considered three different variables.
2. Variable names must start with a letter or an underscore character.
3. Variable names cannot start with a number.
4. Variable names can only contain alphanumeric characters and underscores (A-z, 0-9, and _).
5. Variable names cannot be any of the Python keywords.

6. It is recommended to use lowercase letters for variable names.
7. If a variable name consists of more than one word, use underscores to separate them, such as first_name.
8. Choose descriptive names that reflect the purpose of the variable, such as count instead of c.
9. Use singular nouns for variable names, such as student instead of students.



Examples of Invalid Syntax

Comments in Python

Comments are used to explain code or provide notes within a program. They are ignored by the Python interpreter and are not executed.

There are two types of comments in Python:

1. **Single-line comments:** Start with the hash symbol # and continue until the end of the line.
2. **Multi-line comments:** Enclosed within triple quotes """ or ''', spanning multiple lines.

Strings and String Handling in Python

Strings are used to represent text data in Python. They are enclosed in single or double quotes and can contain letters, numbers, and special characters.

Some common string operations include:

1. **Concatenation:** "Hello " + "World!" => "Hello World!"
2. **Repetition:** "Python " * 3 => "Python Python Python "
3. **Indexing:** "Python" => "P"
4. **Slicing:** "Python"[0:3] => "Pyt"
5. **Built-in methods:** "python".upper() => "PYTHON", " trim ".strip() => "trim"

```
[ ] q='Mandar Janardan'

[ ] q.upper()
'MANDAR JANARDAN'

[ ] q.lower()
'mandar janardan'

[ ] q.title()
'Mandar Janardan'

[ ] q.capitalize()
'Mandar janardan'
```

```
[ ] # indexing,slicing

[ ] q
'Mandar Janardan'

[ ] q[5]
'r'

[ ] q[12]
'd'

[ ] q[-3]
'd'

[ ] s='Automatic'

[ ] s[2:5]
'tom'
```

```
[ ] # var_name[start:end-1:step]

[ ] s[1:7:2]
'uoa'

s[::-1]
'citamotuA'
```

Examples of String Handling

User Input

Users can obtain input using the input function.

Example – name=input('Enter Name')

The Input function can read only strings inputted by the user, but sometimes an integer or float data type needs to be read. In python, if you want to read data types other than Strings you must use a type conversion function.

```
#user Input
score = int(input('Enter Player Score:'))

type(score)
Enter Player Score:89
int

[ ] a=input('Enter Player Score:')
Enter Player Score:56

[ ] a
'56'
```

Type Casting

Casting is converting a variable value from one datatype to another. This is done with functions such as int() or float() or str() in Python. A common method that us used is converting a number, which is a string into a proper number.

Following are the 2 types of type casting:

(a) Implicit Type Conversion

```
[ ]  # Types of Type Conversion

[ ]  # 1 Implicit Type Conversion

[ ]  int_1 = 5
     int_2 = 2
     result = int_1/int_2
     print(result)

     type(result)

     2.5
     float
```

(b) Explicit Type Conversion

```
     # 2 Explicit Type Conversion
     nums = 125
     chars = "Hello"
     numchar = nums + chars

[ ]  numchar = str(nums) + chars
     numchar

     '125Hello'
```

Implicit type conversion happens automatically when possible, like when an integer is added to a float. Explicit type conversion requires using functions like int(), float(), str(), etc.

Operators

1. **Arithmetic Operators (+, -, *, /, %, //,):** Perform mathematical calculations like addition, subtraction, multiplication, etc. Examples: 5 + 3, 10 / 2, 4 ** 2 (exponentiation).

2. **Relational Operators (==, !=, <, >, <=, >=):** Compare values and return True or False based on the relationship. Examples: x == 5 (checks if x equals 5), y > 10 (checks if y is greater than 10).

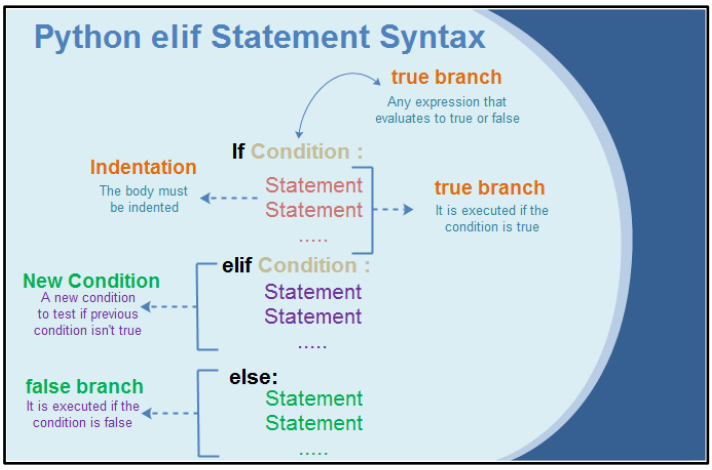
3. **Logical Operators (and, or, not):** Combine conditions. and requires both conditions to be True, or needs at least one True, and not reverses the truth value. Examples: x > 0 and y < 5 (both x and y must meet conditions), name == "Alice" or age == 25 (either name or age must match).

4. **Membership Operators (in, not in):** Check if a value exists within a sequence (list, tuple, string). in returns True if found, not in returns True if not found. Examples: 5 in [1, 2, 3, 5], "apple" not in fruits.

Conditional Statements

Python has several conditional statements:

1. **if:** Executes a block of code if a condition is true.
2. **if-else:** Executes one block if condition is true, else another block.
3. **if-elif-else:** Checks multiple conditions and executes the corresponding block.
4. **Nested if:** An if statement inside another if block.



Iterative Statements / Loops

Python has two main loop statements:

1. **for loop:** Iterates over a sequence (list, tuple, string, etc).
2. **while loop:** Executes a block of code as long as a condition is true.

Functions

Functions are reusable blocks of code that perform a specific task. They can take parameters and return values.

```
[ ]  def add(s,v):
         c=s+v
         print(c)
```

→

```
[ ]  add(5,9)
     14

[ ]  add(9,456)
     465
```

Data Structures

Python provides built-in data structures like lists, dictionaries, sets, and tuples. They allow storing and manipulating collections of data.

Data Handling and Manipulation of Data Frames using Pandas

Pandas is a popular library for data manipulation and analysis. It provides data structures like Series and DataFrame for working with tabular, multidimensional, potentially heterogeneous data.

```
[ ] #Import Python Libraries

    import pandas as pd
```

```
df = pd.read_csv('/content/Salaries (1).csv')
df
```

	rank	discipline	phd	service	sex	salary
0	Prof	B	56.0	49	Male	186960
1	Prof	A	12.0	6	Male	93000
2	Prof	A	23.0	20	Male	110515
3	Prof	A	40.0	31	Male	131205
4	Prof	B	20.0	18	Male	104800

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80 entries, 0 to 79
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   rank        80 non-null     object
 1   discipline  79 non-null     object
 2   phd         79 non-null     float64
 3   service     80 non-null     int64
 4   sex         80 non-null     object
 5   salary      80 non-null     int64
dtypes: float64(1), int64(2), object(3)
memory usage: 3.9+ KB
```

```
[ ] df.shape
```

```
(80, 6)
```

```
[ ] df.size
```

```
480
```

Selecting a column in a Data Frame

Method 1: Subset the data frame using column name: df['phd']

Method 2: Use the column name as an attribute: df.phd

```
[ ] xyz = df[['phd','service','rank']]
```

```
[ ] a=df[['service','rank']]
```

```
a.head()
```

	service	rank
0	49	Prof
1	6	Prof
2	20	Prof
3	31	Prof
4	18	Prof

Data Frames groupby method

Using "group by" method we can:

- Split the data into groups based on some criteria
- Calculate statistics (or apply a function) to each group
- Similar to dplyr() function in R

```
[ ] #Group data using rank
    df_rank = df.groupby('rank')
```

```
[ ] #Calculate mean value for each numeric column per each group
    df_rank['salary'].mean()
```

```
rank
AssocProf     91786.230769
AsstProf      81362.789474
Prof         123347.895833
Name: salary, dtype: float64
```

Creating and Manipulating Matrices using NumPy

NumPy is a library for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices. It also includes a large collection of high-level mathematical functions to operate on these arrays.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
[ ] a=np.array([1,2,3,45,5])
```

```
a
array([ 1,  2,  3, 45,  5])
```

```
[ ] a[3]
45
```

```
[ ] a[1:4]
array([ 2,  3, 45])
```

```
[ ] type(a)
numpy.ndarray
```

```
[ ] q=np.arange(1,50)
```

```
[ ] q
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])
```

```
[ ] r=np.arange(27).reshape(3,3,3)
```

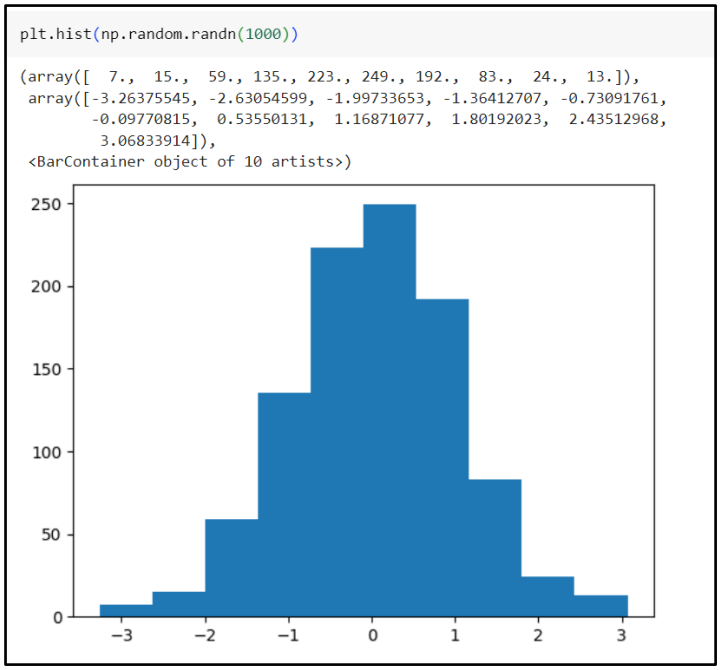
```
[ ] r
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

Visualizing Data using EDA (Exploratory Data Analysis) and Matplotlib

Matplotlib is a plotting library for Python, producing publication-quality figures in a variety of hardcopy formats and interactive environments. It is commonly used for exploratory data analysis (EDA) to gain insights from data.

```
plt.hist(np.random.randn(1000))
```

```
(array([  7.,  15.,  59., 135., 223., 249., 192.,  83.,  24.,  13.]),
 array([-3.26375545, -2.63054599, -1.99733653, -1.36412707, -0.73091761,
        -0.09770815,  0.53550131,  1.16871077,  1.80192023,  2.43512968,
         3.06833914]),
 <BarContainer object of 10 artists>)
```



CHAPTER 4 – SQL (STRUCTURED QUERY LANGUAGE)

Mentor: Mr. Mandar Katkar (Data Science Trainer)

SQL stands for Structured Query Language. It is a computer language used for storing and managing, manipulating and retrieving data in relational database management system (RDMS). It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.

All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language. SQL can be integrated with other programming languages, for instance, embedding SQL queries with the Java programming language to build high – performing data processing applications.

SQL standards are a set of formally defined guidelines of the Structured Query Language (SQL). The American National Standards Institute (ANSI) and the International Organization of Standardization (ISO) adopted the SQL standards in 1986.

Components of an SQL system –

1. **SQL Table –** It is the basic element of a relational database which consists of rows and columns representing different entities, their data attributes and the various relationships between the data values.

2. **SQL Statements** – Also known as SQL Queries, these statements are valid instructions that relational database management system understand. The SQL Language elements that constitute a correct SQL statement include – operators, identifiers, variables and their data types, and search conditions.

3. **Stored Procedures** – They are a collection of one or more SQL statements stored in the relational database used to improve the efficiency and performance of the overall SQL Relational Database System.

Advantages of SQL –

1. **Faster and Efficient Query Processing –** SQL can quickly and efficiently retrieve a large volume of data records from a database. Compared to an unstructured database such as MongoDB, it is a relational database that can characterize the data in a structured way.

   Operations like – insertion, deletion, querying, manipulation and calculations – on data via analytical queries in a relational database can be accomplished in a matter of seconds.

2. **Portable –** SQL is highly portable because it is employed in programs on PCs, servers, tablets and independent laptops running operating systems such as Windows, Linux, Mac and even some mobile phones. It can also be embedded with other programs based on the requirements.

3. **Standardized language –** It gives all users a consistent platform worldwide due to proper documentation, making it convenient for developers and businesses to access and shift to different databases.

4. **Data Integrity –** SQL ensures the consistency and accuracy of data by using constraints such as primary key, foreign key, NOT NULL and UNIQUE constraints.
   Following are the 4 types of data integrity maintained by SQL –
   (a) Entity Integrity – Ensures that there are no duplicate rows in a table.
   (b) Domain Integrity – Enforces valid entries for a given column by restricting the data type.
   (c) Referential Integrity – Rows cannot be deleted which are used by other records.
   (d) User – defined Integrity – enforces some specific business rules.

5. **Flexibility –** SQL, being a versatile language, empowers users to perform complex data analysis and management tasks efficiently.

Disadvantages of SQL –

1. **Resource Intensive Scaling –** SQL databases typically scale up vertically by increasing hardware investment which is both costly and time – consuming.
2. **Partial control –** SQL does not provide programmers complete control over databases primarily due to hidden corporate rules.
3. **Cost inefficient –** some versions of SQL are expensive, making it unaccessible to programmers.
4. **Rigidity –** The schema of a SQL database must be specified before it can be used. They are rigid once installed, and changes are often complex and time – consuming, hence limiting the ability of developers and businesses to customize it according to their needs.
5. **Security threats –** Certain security threats / risks such as injection attacks may further compromise the integrity and confidentiality of data stored.

Data Types in SQL –

Category	Data Type	Syntax	Description
String	Character	CHAR	It stores string values containing any characters in a character set. It is defined to be of fixed length.
	Varying Character	VARCHAR VARCHAR2	It stores string values containing variable characters in a set of characters but of defined variable length.
	Binary Large Object	BLOB	It stores binary string values in hexadecimal format. It has a defined variable length.
Numeric	Numeric	NUMERIC NUMBER	It stores exact numbers with a defined precision and scale. It has decimals (similar to the datatype float in python).
	Integer	INT	It stores exact numbers with a pre – defined precision and scale to zero. It stores integer values.
Temporal	Timestamp	TIMESTAMP	It stores a moment an event occurs, using a definable fraction – of – a second precision.
	Timestamp with local	TIMESTAMP WITH LOCAL	It stores a moment an event occurs, using a definable fraction – of – a second precision, with respect to the local time zone.
	Timezone	TIMEZONE	It defines the timezone.
Boolean	Boolean	BOOLEAN	It checks the condition and stores the following values – True, False or Unknown.

SQL Commands –

Following are the 4 types of SQL Commands –

1. **Data Definition Language (DDL) –** Adding or deleting the table
2. **Data Manipulation Language (DML) –** Changing or inserting or replacing the data / table values
3. **Data Control Language (DCL) –** Giving a set of controls / rights / access to the group of users for a particular table
4. **Data Query Language (DQL) –** Stored data can be retrieved using a query and can be displayed / shown to the users.

Type of SQL Command	Command	Description	Syntax
Data Definition Language (DDL)	CREATE	It creates a database, new tables, a view of the table or other object in the database.	CREATE DATABASE *database_name*; CREATE TABLE *table_name* (*column1_name datatype, column2_name datatype*, …);
	ALTER	It modifies / alters an existing database object, such as the table.	ALTER TABLE *table_name* ADD *column_name datatype*;
	DROP	It is deletes a database, an entire table, views of the table or other object in the database.	DROP DATABASE *database_name;* DROP TABLE *table_name*;
Data Manipulation Language (DML)	INSERT	It inserts a new record in a table.	INSERT INTO *table_name* (*column1_name, column2_name*) VALUES (*value1, value2*);
	UPDATE	It modifies an existing record.	UPDATE *table_name* SET *column_name = value* WHERE *condition*;

	DELETE	It deletes an existing record in a table.	DELETE FROM *table_name* WHERE *condition*;
	COMMIT	It is used to commit a transaction.	COMMIT;
	ROLLBACK	It is used to rollback a transaction in case any error occurs.	ROLLBACK;
Data Control Language (DCL)	GRANT	It assigns privileges to a user to access the database.	GRANT *priviledge_name* ON *database_*name.table*_name* TO *username*;
	REVOKE	It used to take back the privileges granted to a user for a particular database.	REVOKE *priviledge_name* ON *database_name.table_name* FROM *username*;
Data Query Language (DQL)	SELECT	It is used to retrieve certain records from one or more tables and display it in the console.	SELECT *column1_name, column2_name…* FROM *table_name*;
	SELECT DISTINCT	It is used to return only distinct (different) values of certain records from one or more tables and display it in the console.	SELECT DISTINCT *column1_name, column2_name…* FROM *table_name*;
	WHERE	It is a clause used to filter records based on a condition.	SELECT *column1_name*, *column2_name*,…. FROM *table_name*;

	HAVING	It is a clause used with aggregate functions to filter records based on a condition.	SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)* HAVING *condition* ORDER BY *column_name(s)*;
	IN	It is an operator used to specify multiple values in a WHERE clause. It is used to retrieve only those records from a table that have the specified value.	SELECT *column1_name, column2_name,…* FROM *table_name* WHERE *column_name* IN (*value1, value2….*);
	NOT IN	It is an operator used to specify multiple values in a WHERE clause. It is used to retrieve records from a table except for those records that have the specified value.	SELECT *column1_name, column2_name,…* FROM *table_name* WHERE *column_name* NOT IN (*value1, value2….*);
	IS NULL	It checks if the value stored in a record of a table is null.	SELECT *column1_name*, *column2_name,…* FROM *table_name* WHERE *column_name* IS NULL;
	IS NOT NULL	It checks if the value stored in a record of a table is not null.	SELECT *column1_name*, *column2_name,…* FROM *table_name* WHERE *column_name* IS NOT NULL;
	ORDER BY	It is used to sort the selected record set in either ascending	SELECT *column1_name, column2_name,…*

		(ASC) or descending (DESC) order.	FROM *table_name* WHERE *condition* ORDER BY *column_name* {ASC | DESC};
	LIKE	It is an operator used in a WHERE clause used to search for a specified pattern in a column. Underscore (_) indicates a single character. Percent sign (%) indicates more than one characters.	SELECT *column1_name, column2_name,…* FROM *table_name* WHERE *column_name* LIKE *pattern*; -- SELECT * FROM CUSTOMERS WHERE CUSTOMER_NAME LIKE '_a%';
	GROUP BY	It is used with aggregate functions [COUNT( ), MAX( ), MIN( ), SUM( ), AVG( )] to group the required result set by one or more columns.	SELECT *column1_name, column2_name,…* FROM *table_name* WHERE *condition* GROUP BY *column_name*;
	COUNT( )	It returns the number of rows that matches a specified criterion.	SELECT COUNT (*column_name*) AS *column_name_to_be_displayed* FROM *table_name* WHERE *condition*;
	MIN( )	It returns the smallest value of the selected column.	SELECT MIN (*column_name*) AS *column_name_to_be_displayed* FROM *table_name* WHERE *condition*;

	MAX( )	It returns the largest value of the selected column.	SELECT MAX (*column_name*) AS *column_name_to_be_displayed* FROM *table_name* WHERE *condition*;
	SUM( )	It returns the total sum of a numeric column.	SELECT SUM (*column_name*) AS *column_name_to_be_displayed* FROM *table_name* WHERE *condition*;
	AVG( )	It returns the average value of a numeric column.	SELECT AVG (*column_name*) AS *column_name_to_be_displayed* FROM *table_name* WHERE *condition*;

Date – 11/05/2024

Outputs for – Day 2 Assignment

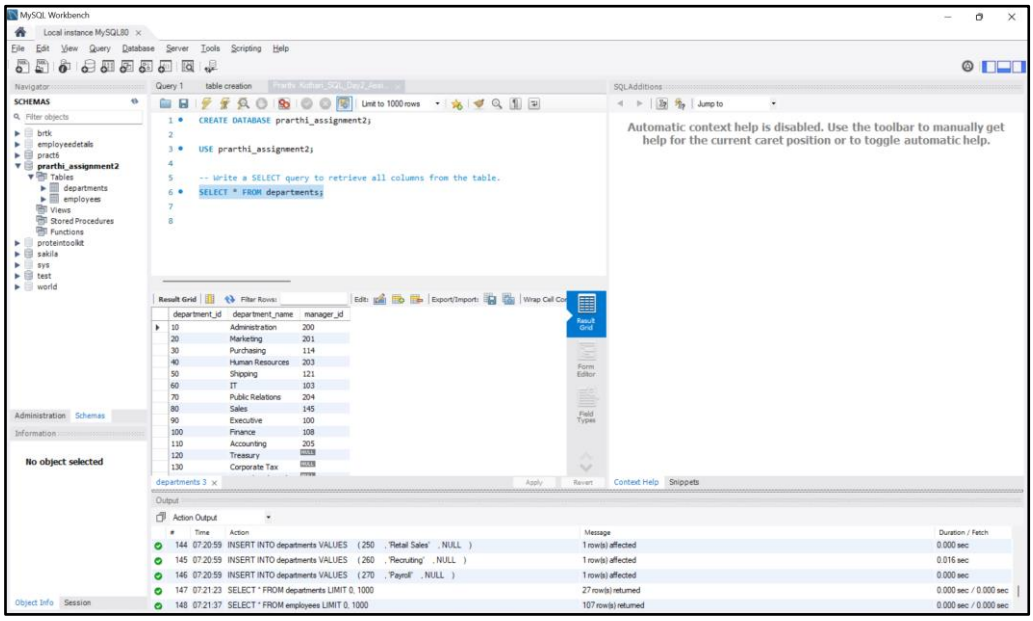
Question 1 –

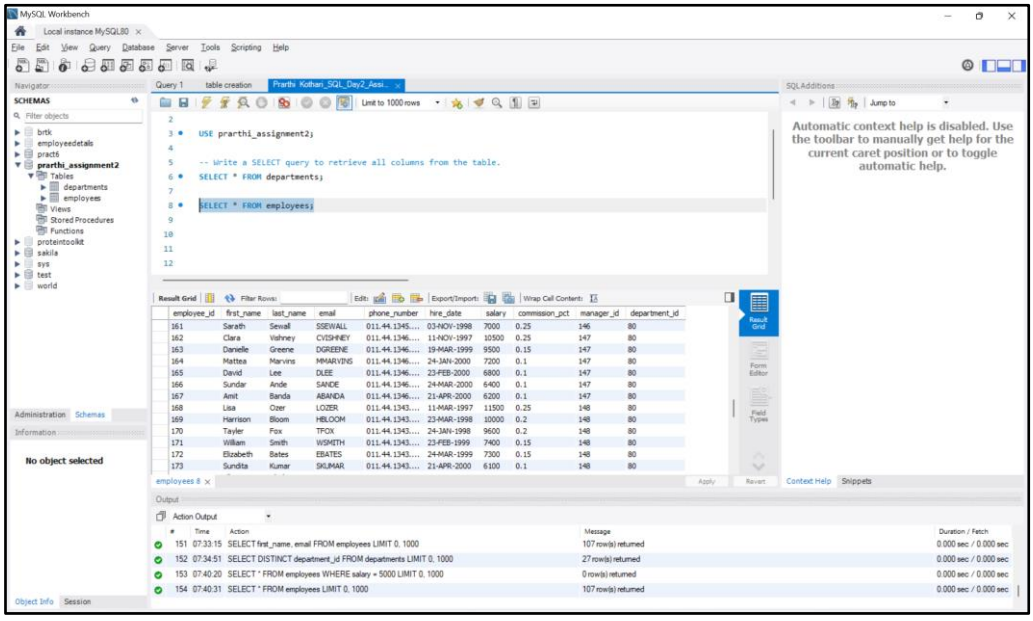
Write a SELECT query to retrieve all columns from the table.

Code –

```
SELECT * FROM departments;
SELECT * FROM employees;
```

Output –



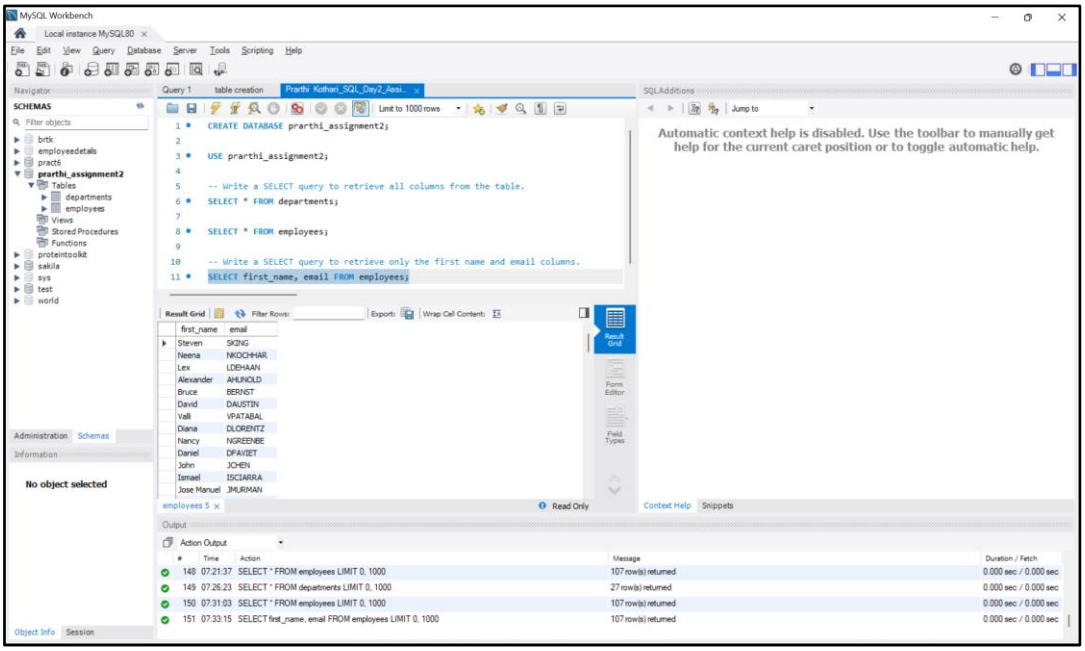


Question 2 –
Write a SELECT query to retrieve only the first name and email columns.

Code –

```
SELECT first_name, email FROM employees;
```

Output –

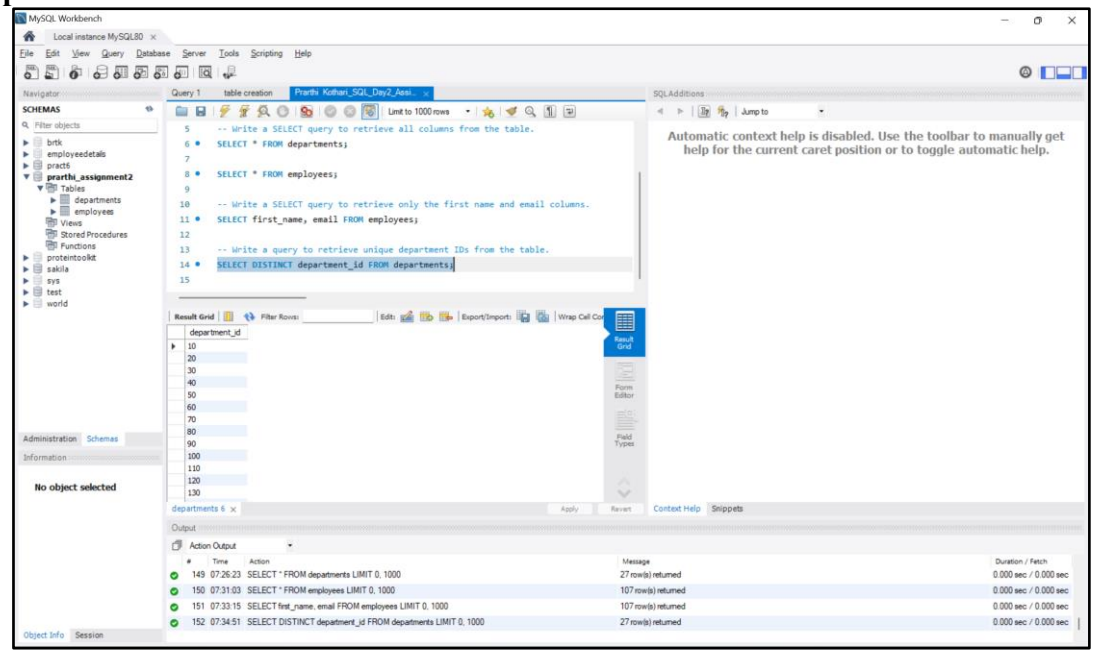


Question 3 –
Write a query to retrieve unique department IDs from the table.

Code –

```
SELECT DISTINCT department_id FROM departments;
```

Output –

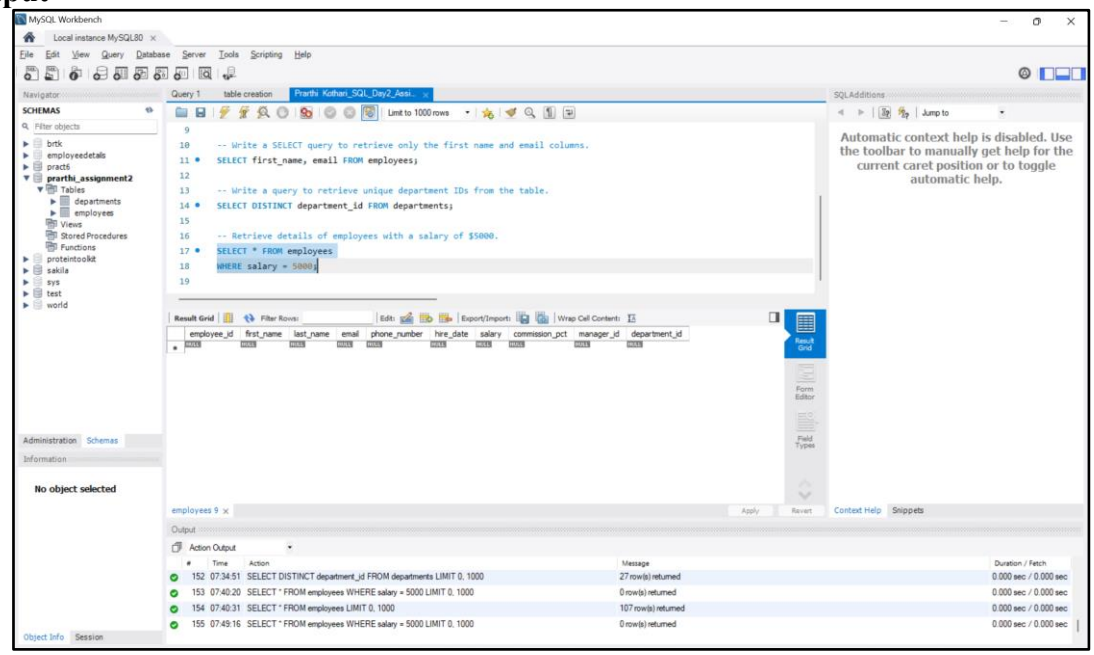


Question 4 –
Retrieve details of employees with a salary of $5000.

Code –

```
SELECT * FROM employees WHERE salary = 5000;
```

Output –

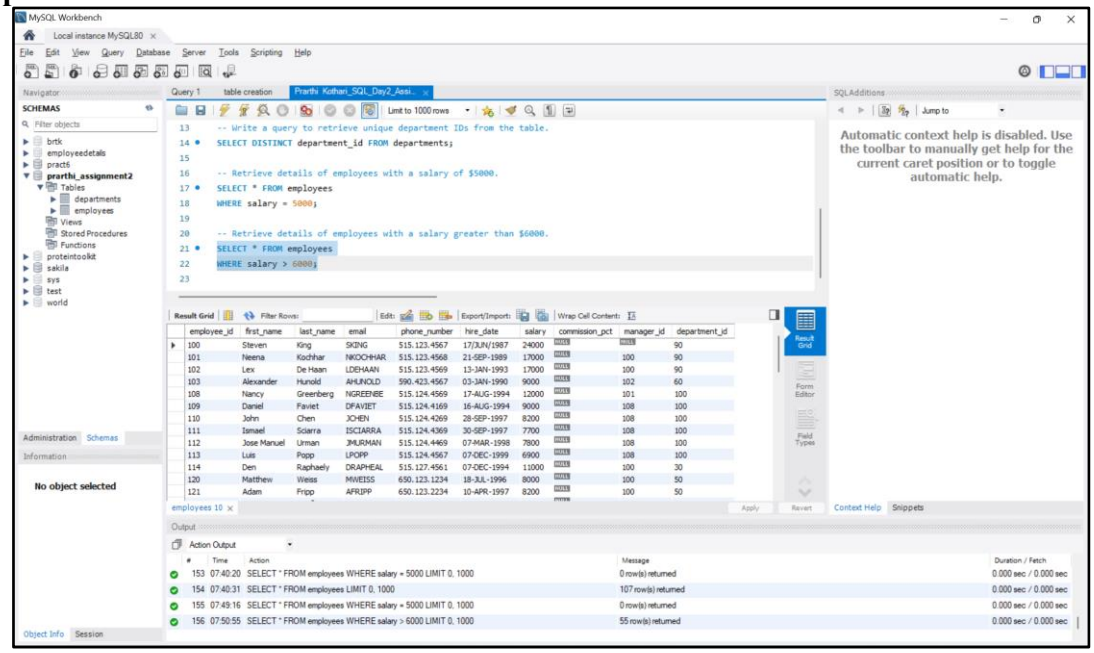


Question 5 –
Retrieve details of employees with a salary greater than $6000.

Code –

```
SELECT * FROM employees WHERE salary > 6000;
```

Output –

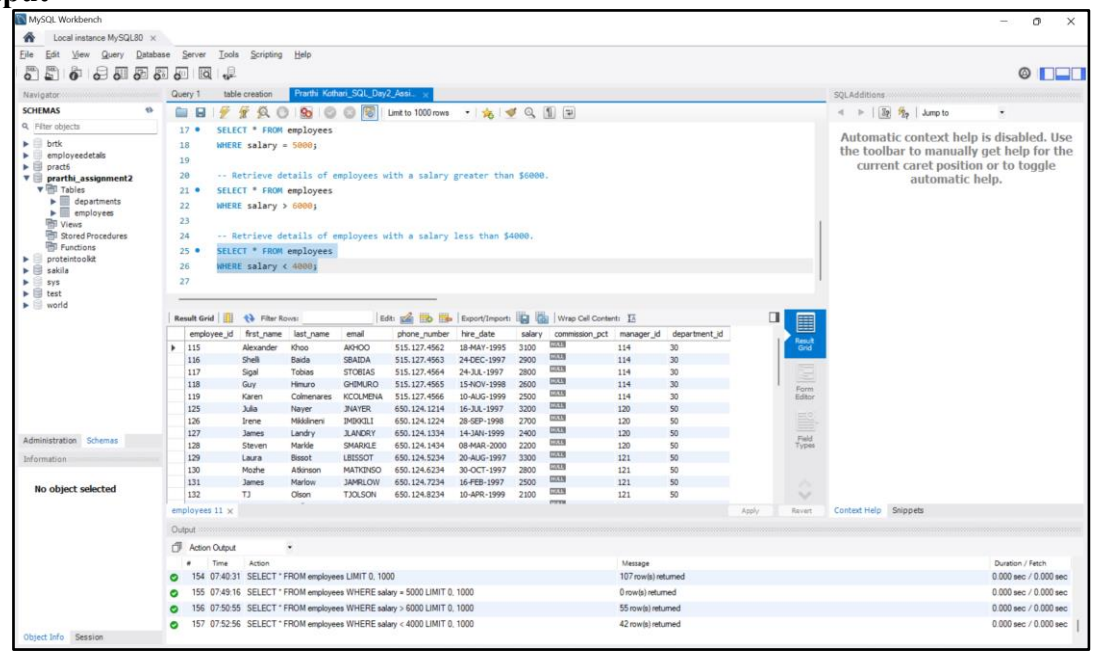


Question 6 –
Retrieve details of employees with a salary less than $4000.

Code –

```
SELECT * FROM employees WHERE salary < 4000;
```

Output –

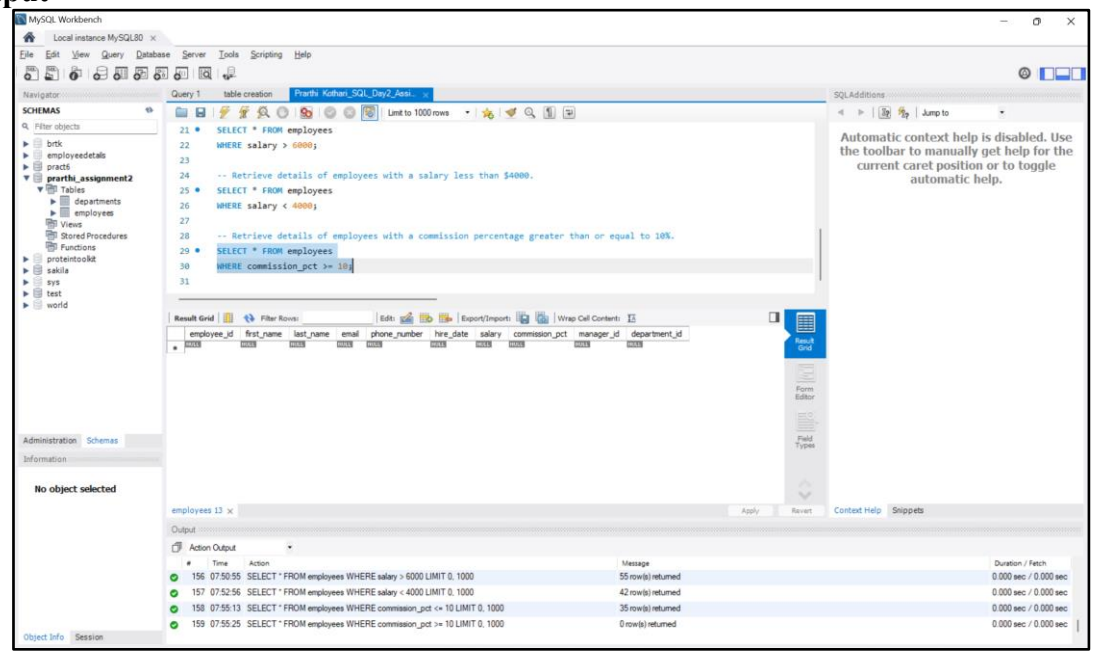


Question 7 –
Retrieve details of employees with a commission percentage greater than or equal to 10%.

Code –

```
SELECT * FROM employees WHERE commission_pct >= 10;
```

Output –

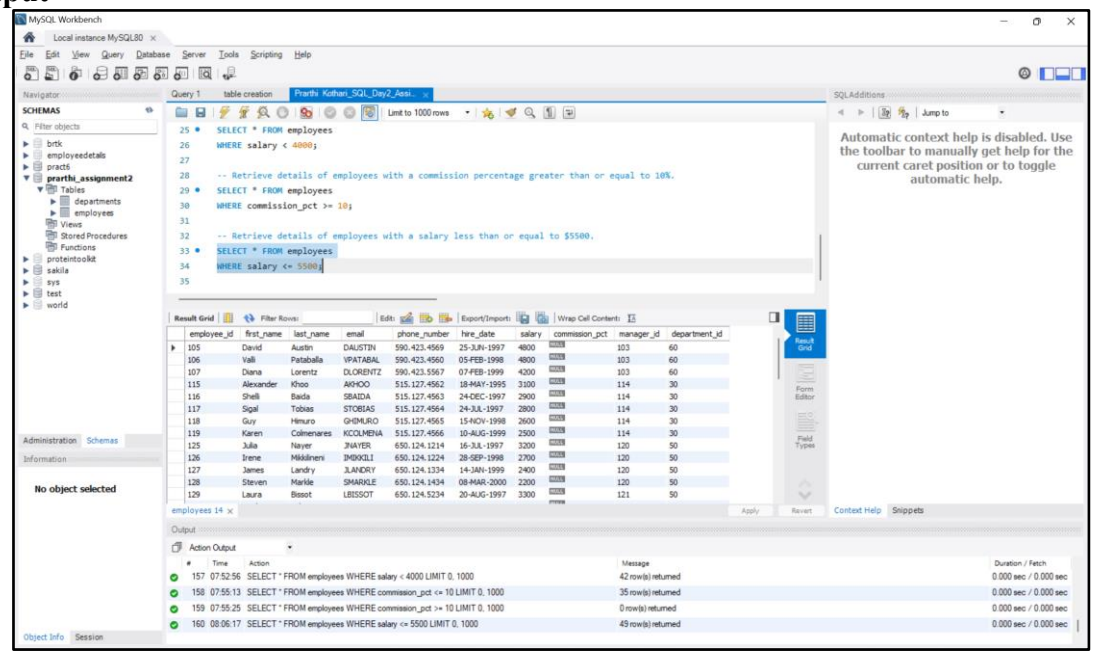


Question 8 –
Retrieve details of employees with a salary less than or equal to $5500.

Code –

```
SELECT * FROM employees WHERE salary <= 5500;
```

Output –

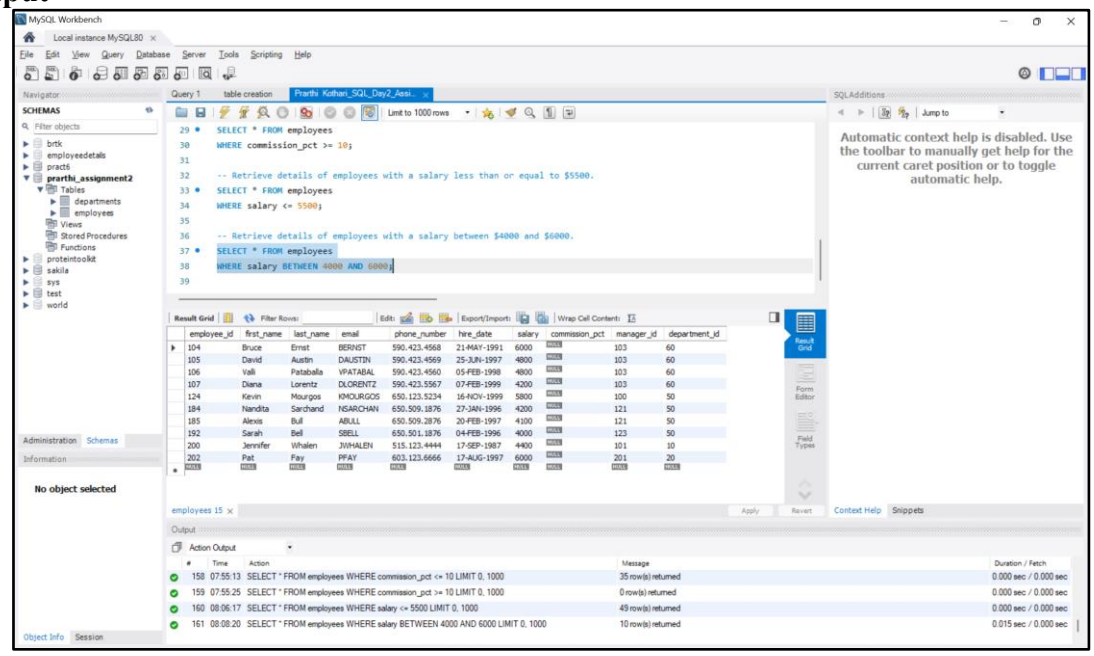


Question 9 –
Retrieve details of employees with a salary between $4000 and $6000.

Code –

```
SELECT * FROM employees WHERE salary BETWEEN 4000 AND 6000;
```

Output –

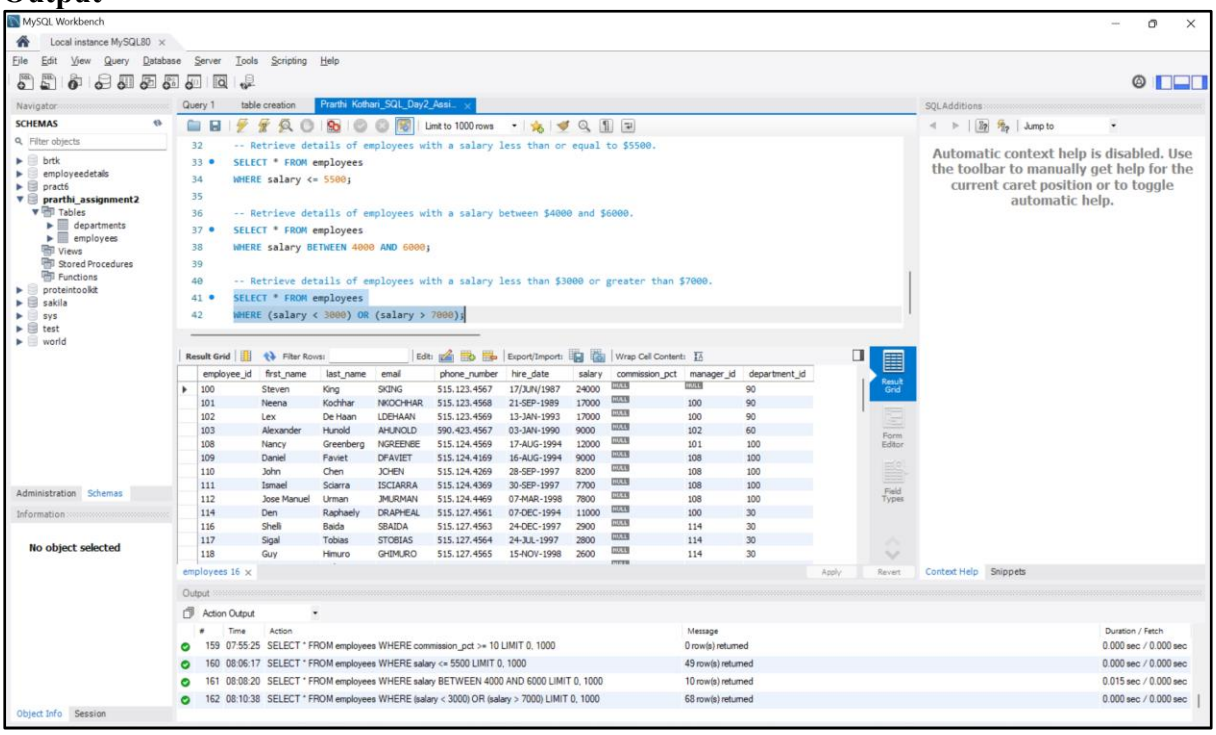


Question 10 –
Retrieve details of employees with a salary less than $3000 or greater than $7000.

Code –

```
SELECT * FROM employees
WHERE (salary < 3000) OR (salary > 7000);
```

Output –



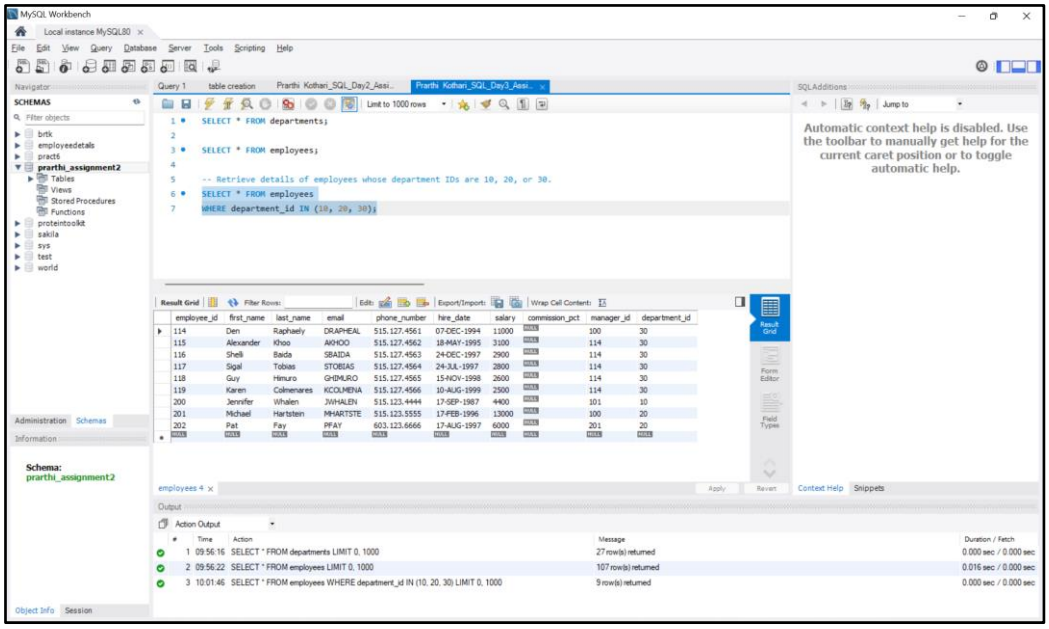
Date – 13/05/2024

Outputs for – Day 3 Assignment

Question 1 –
Retrieve details of employees whose department IDs are 10, 20, or 30.

Code – SELECT * FROM employees WHERE department_id IN (10, 20, 30);

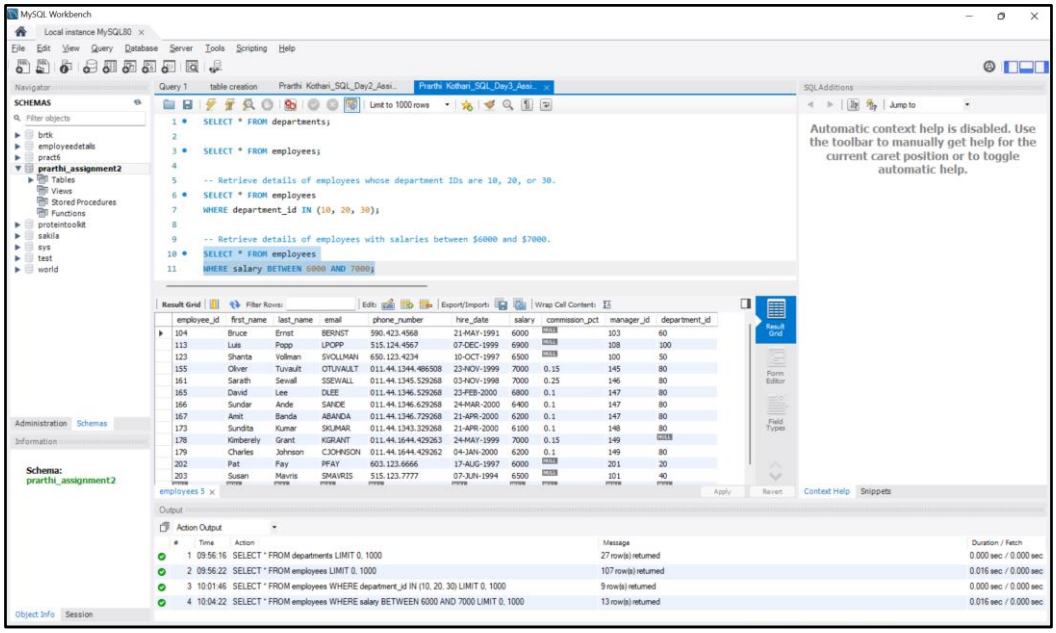
Output –



Question 2 –
Retrieve details of employees with salaries between $6000 and $7000.

Code – SELECT * FROM employees WHERE salary BETWEEN 6000 AND 7000;

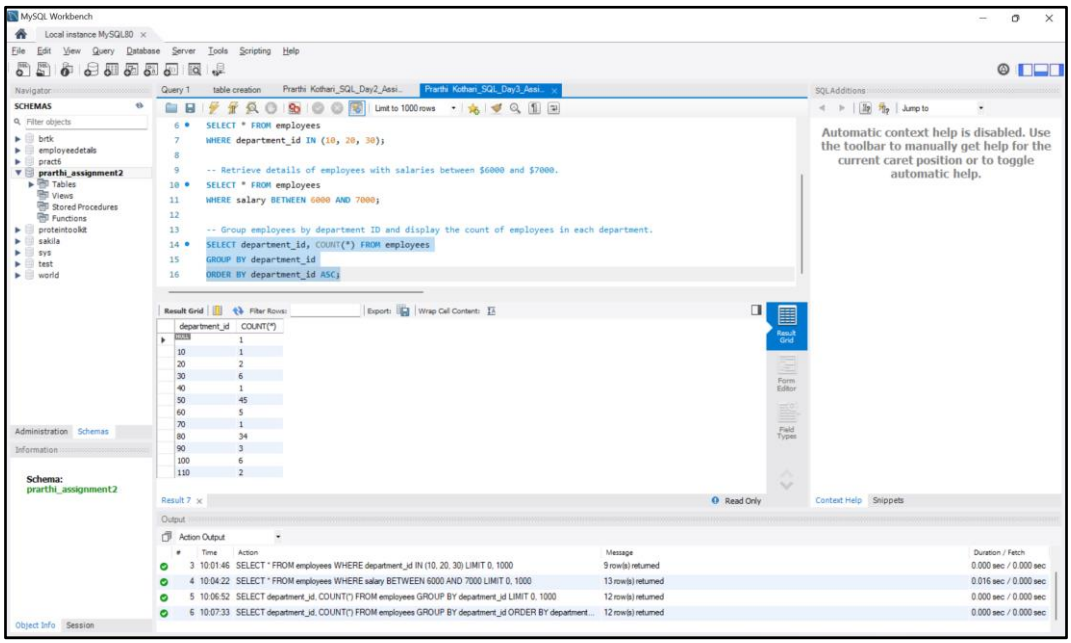
Output –



Question 3 –
Group employees by department ID and display the count of employees in each department.

Code – SELECT department_id, COUNT(*) FROM employees
GROUP BY department_id
ORDER BY department_id ASC;

Output –

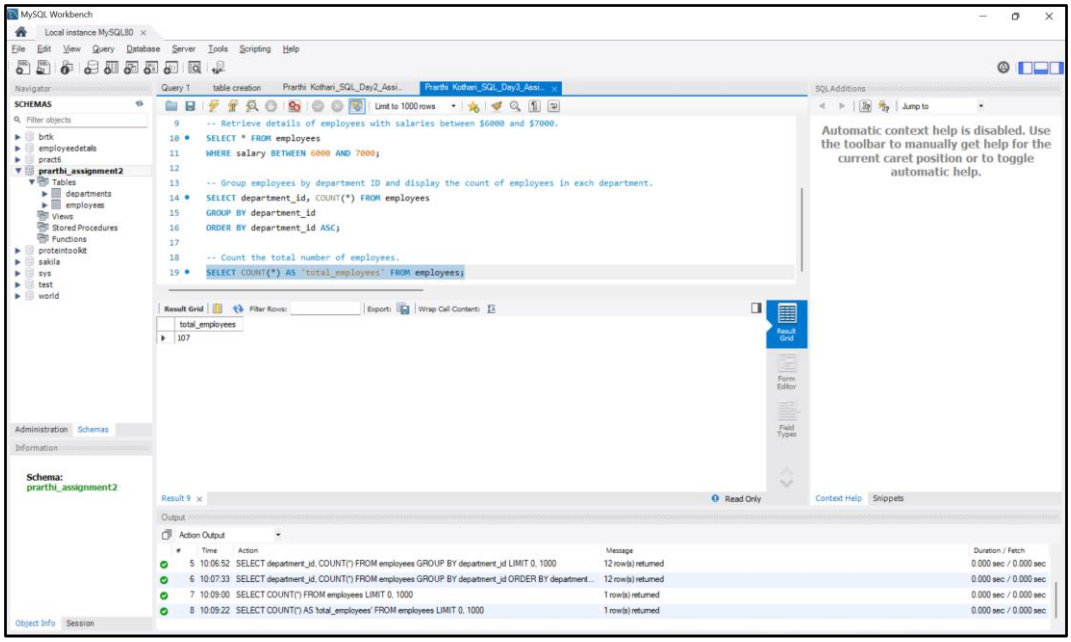


Question 4 –
Count the total number of employees.

Code –

SELECT COUNT(*) AS 'total_employees' FROM employees;

Output –

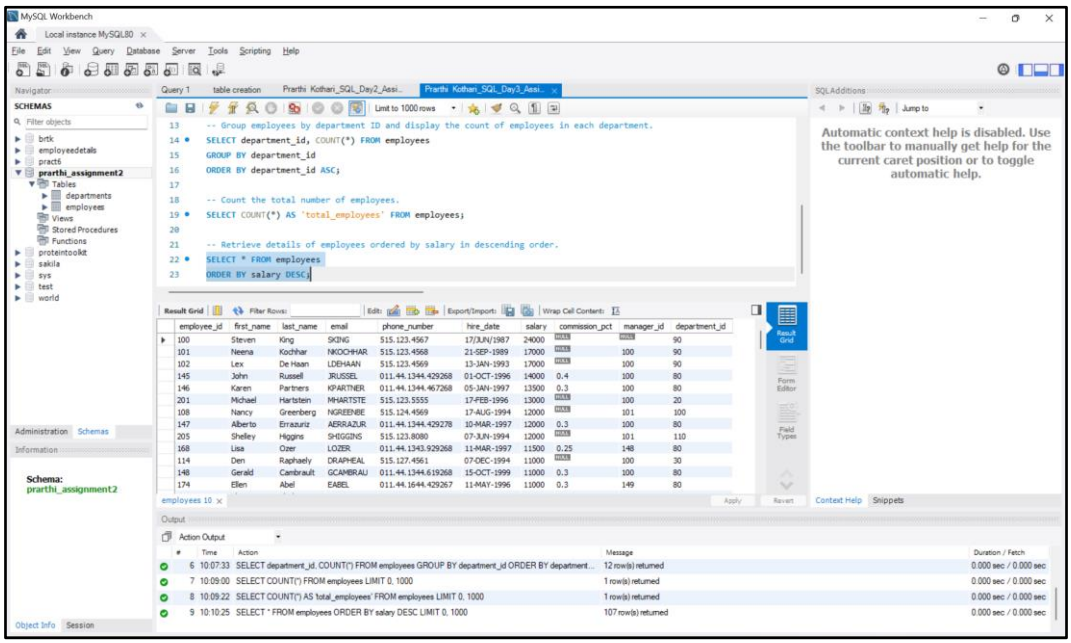


Question 5 –
Retrieve details of employees ordered by salary in descending order.

Code –

```
SELECT * FROM employees
ORDER BY salary DESC;
```

Output –

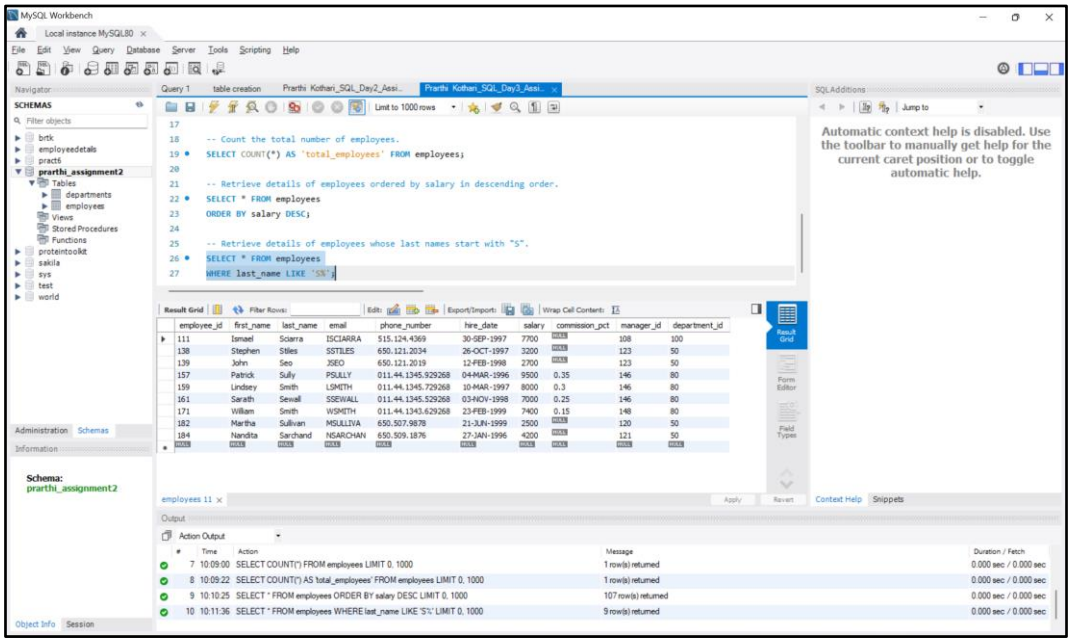


Question 6 –
Retrieve details of employees whose last names start with "S".

Code –

```
SELECT * FROM employees WHERE last_name LIKE 'S%';
```

Output –



Date – 15/05/2024

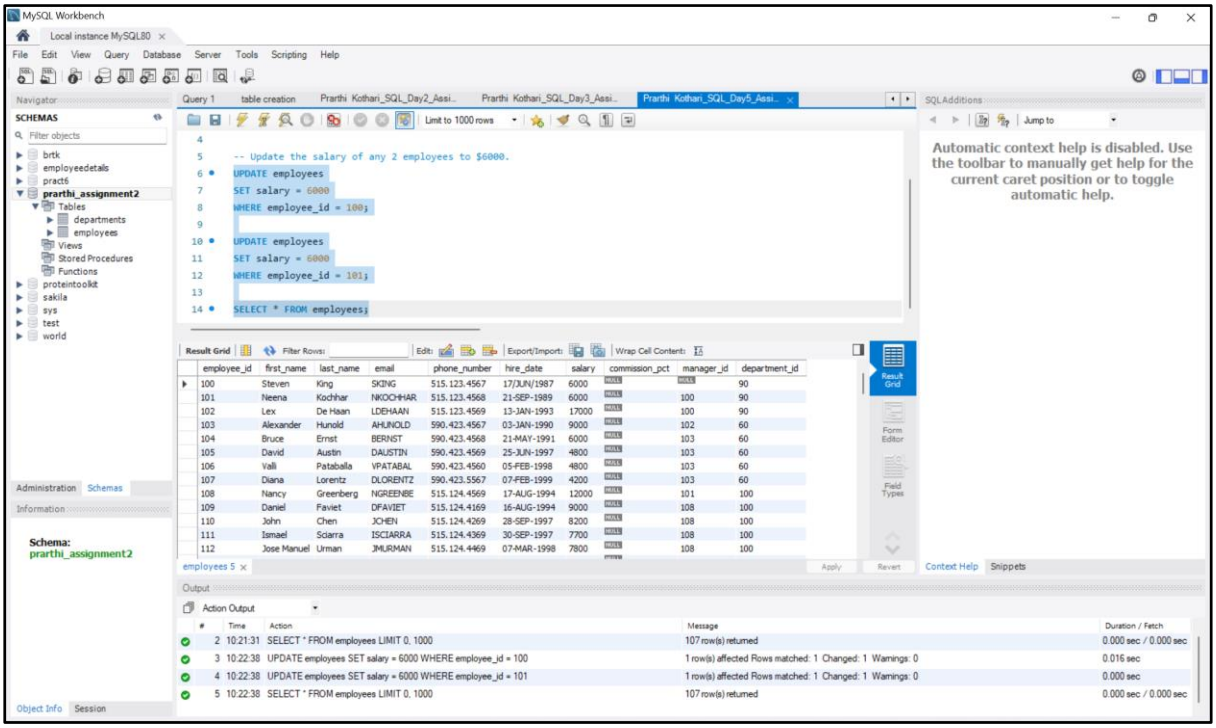
Outputs for – Day 5 Assignment

Question 1 –
Update the salary of any 2 employees to $6000.

Code –

```
UPDATE employees
SET salary = 6000
WHERE employee_id = 100;

UPDATE employees
SET salary = 6000
WHERE employee_id = 101;

SELECT * FROM employees;
```

Output –

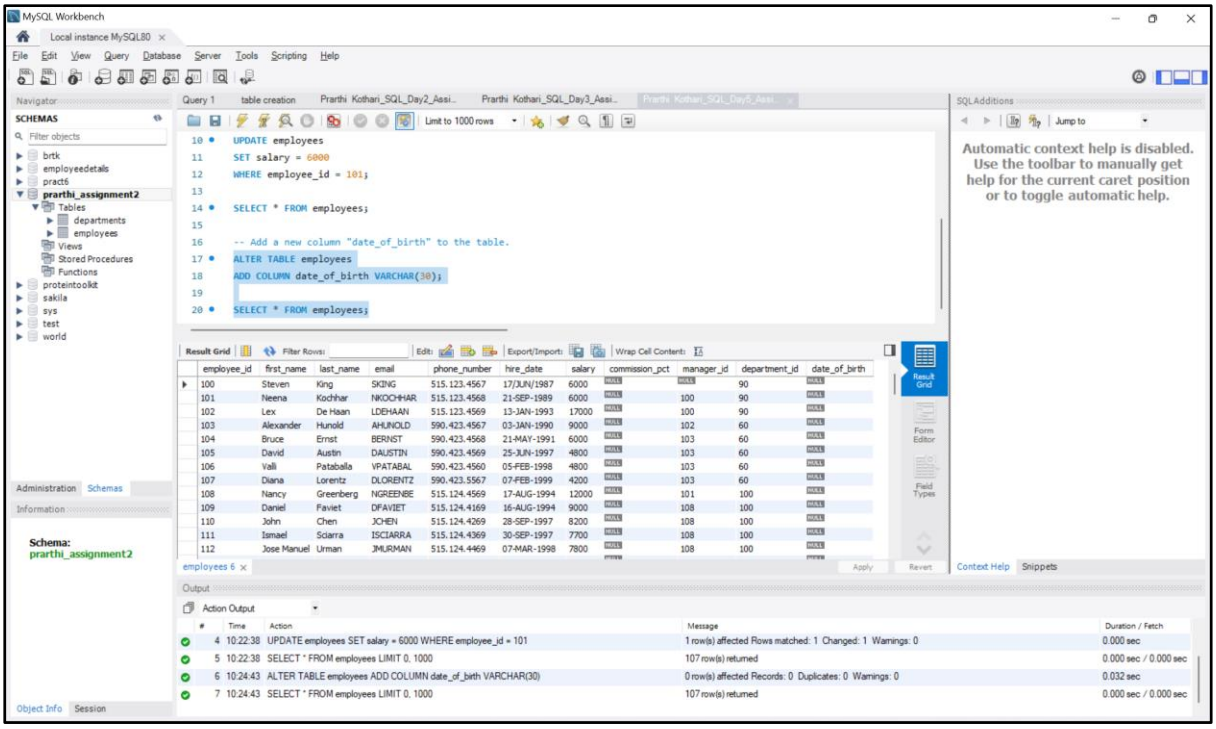


Question 2 –
Add a new column "date_of_birth" to the table.

Code –

```
ALTER TABLE employees
ADD COLUMN date_of_birth VARCHAR(30);

SELECT * FROM employees;
```

Output –

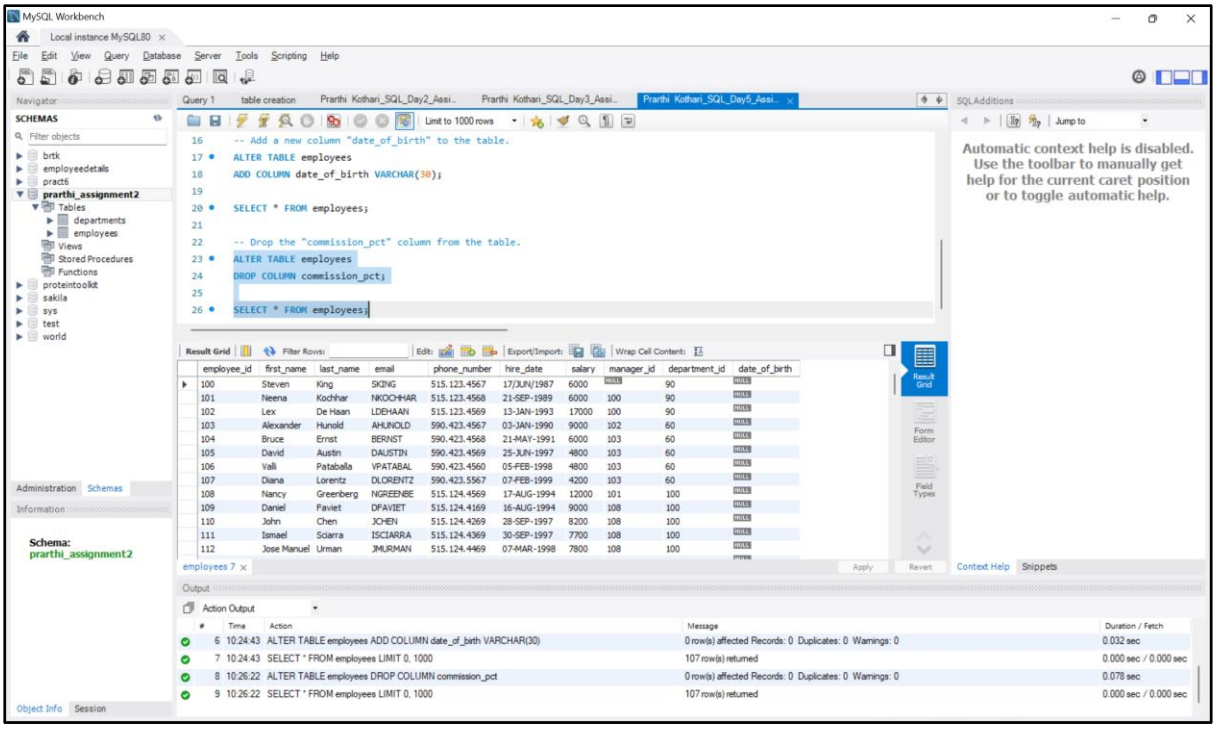


Question 3 –
Drop the "commission_pct" column from the table.

Code –

```
ALTER TABLE employees
DROP COLUMN commission_pct;

SELECT * FROM employees;
```

Output –

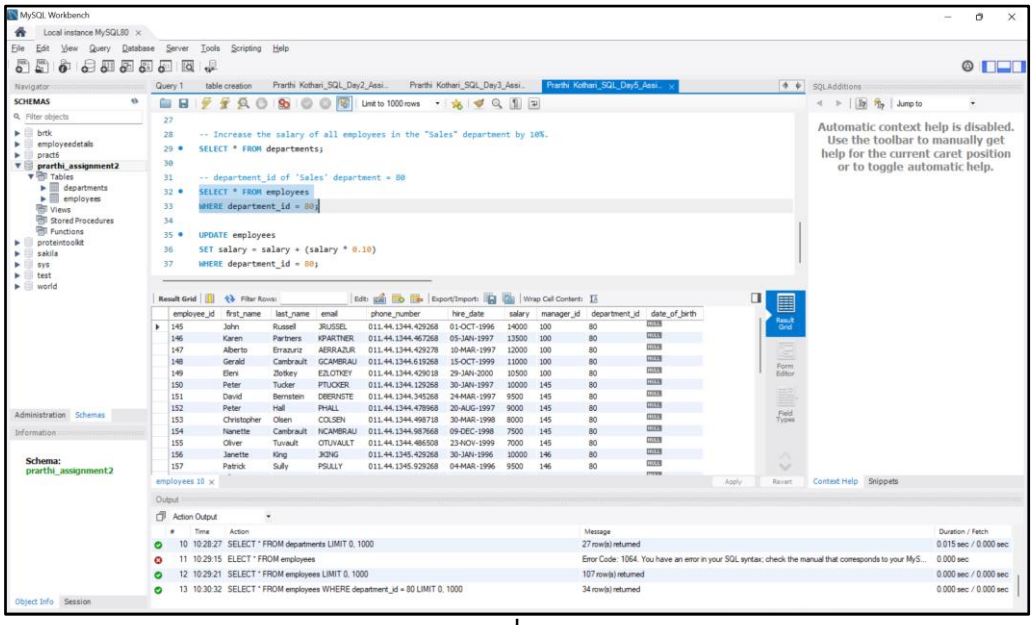


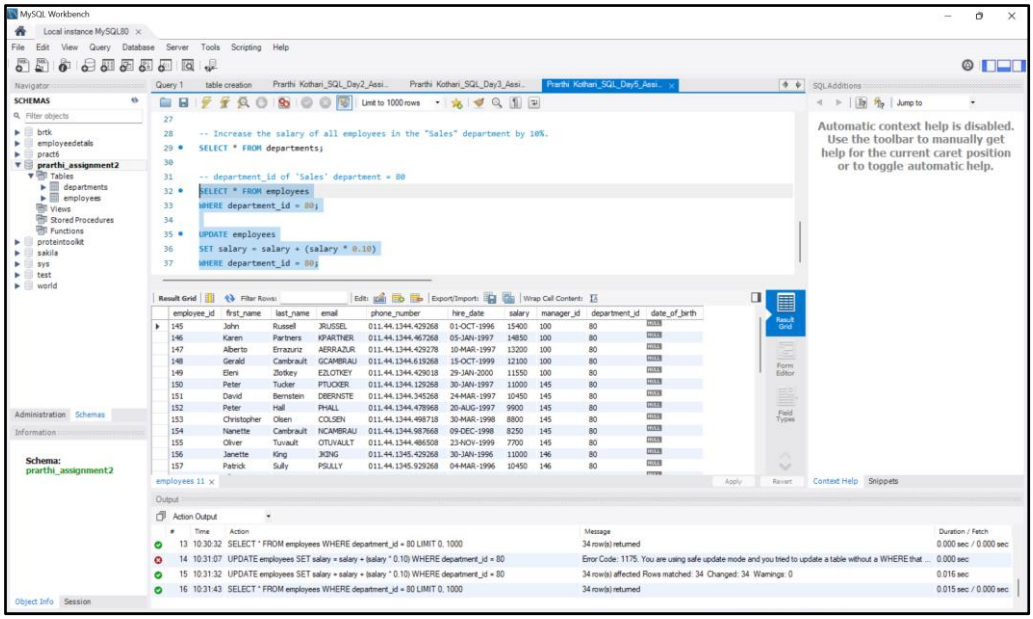
Question 4 –
Increase the salary of all employees in the "Sales" department by 10%.

Code –

```
-- department_id of 'Sales' department = 80
SELECT * FROM employees
WHERE department_id = 80;

UPDATE employees
SET salary = salary + (salary * 0.10)
WHERE department_id = 80;
```

Output –



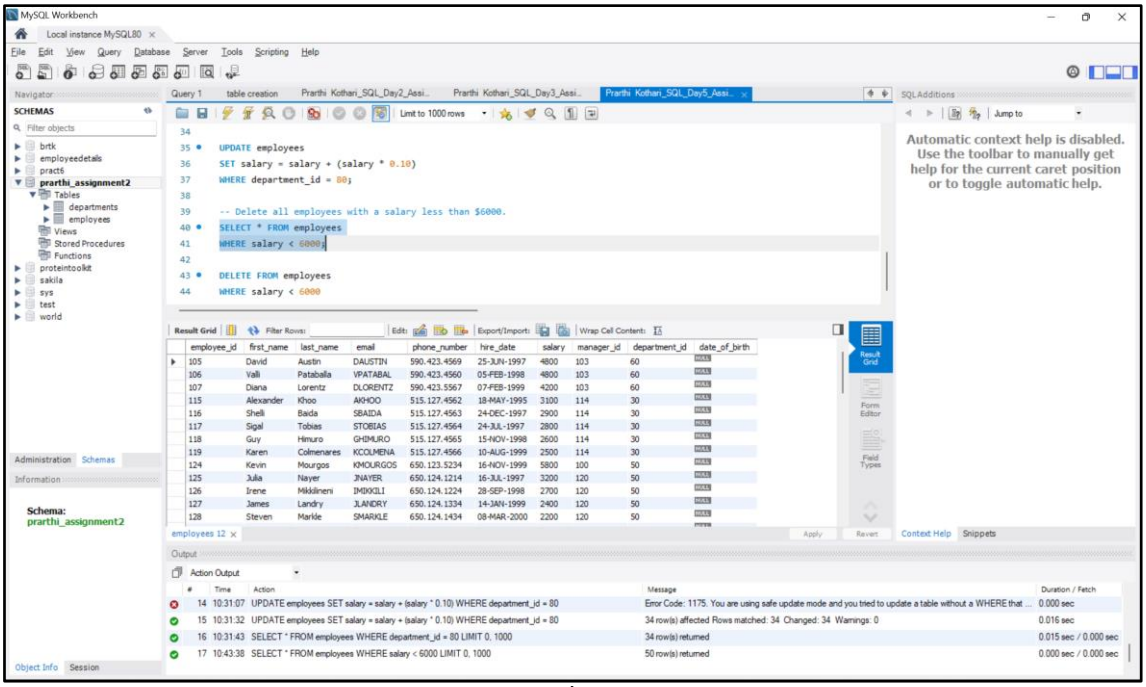


Question 5 –
Delete all employees with a salary less than $6000.

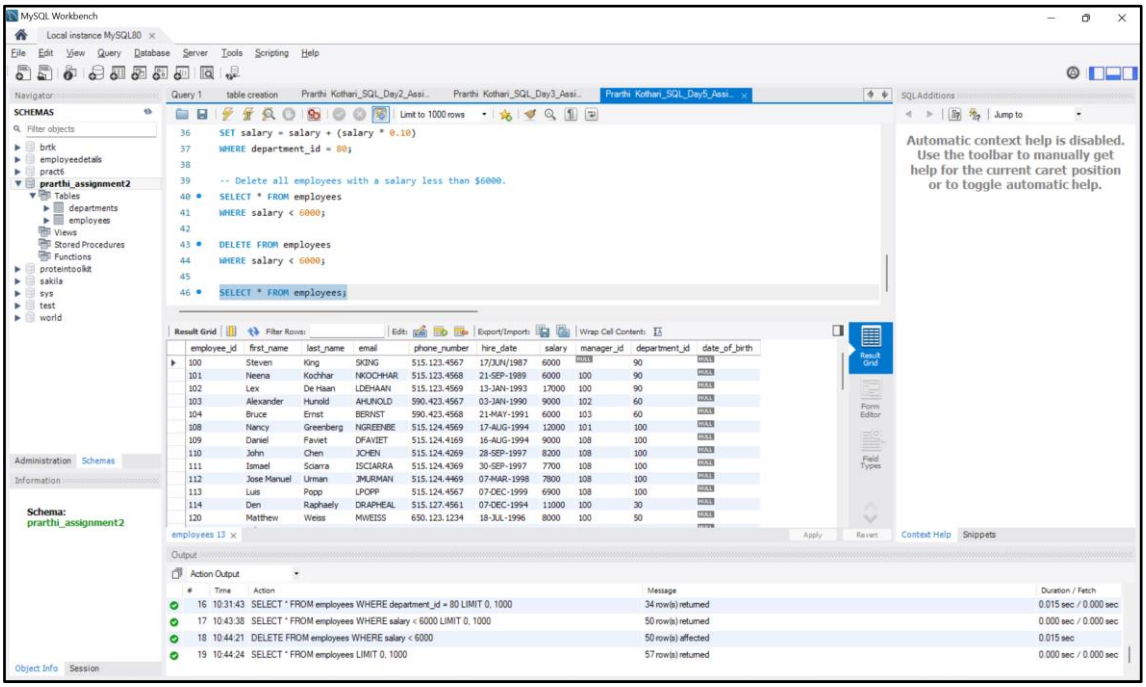
Code –

```
DELETE FROM employees
WHERE salary < 6000;

SELECT * FROM employees;
```

Output –



↓



CHAPTER 5 – ADVANCED EXCEL

Mentor: Mrs. Sushma Ghadge (MIS and Graphics Teacher)

MS Excel is a powerful spreadsheet program developed by Microsoft as part of its Office suite of productivity software. It is widely used in business and enterprise settings for organizing, analyzing, and visualizing data.

Some key features of MS Excel include:

1. A grid-like interface with rows and columns that form cells, where data can be entered and manipulated
2. Formulas and functions for performing calculations, data analysis, and creating charts and graphs
3. Advanced data management tools like sorting, filtering, and pivot tables
4. Compatibility with other Office applications and the ability to import/export data in various formats
5. Programming support through Visual Basic for Applications (VBA) macros

MS Excel was first released for Macintosh in 1985 and then for Windows in 1987, quickly becoming a dominant player in the spreadsheet software market. It has since been continually updated and expanded with new features over the years, making it an essential tool for data-driven tasks across many industries.

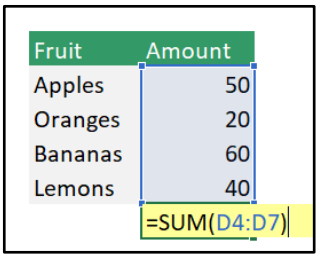
The latest versions of MS Excel are part of the Microsoft Office 2019 and Office 365 suites, providing users with a modern, feature-rich spreadsheet application.

Steps to perform the following functionalities of MS-Excel

1. **Adding Numerical Values**
   - To add numbers in Excel, you can use formulas or the SUM function:

   (a) Formulas always start with an equals sign (=) followed by the numbers and operators. For example, to add 2 and 3, enter =2+3.

   (b) The SUM function adds a range of cells. For example, =SUM(A2:A10) adds the values in cells A2 through A10.



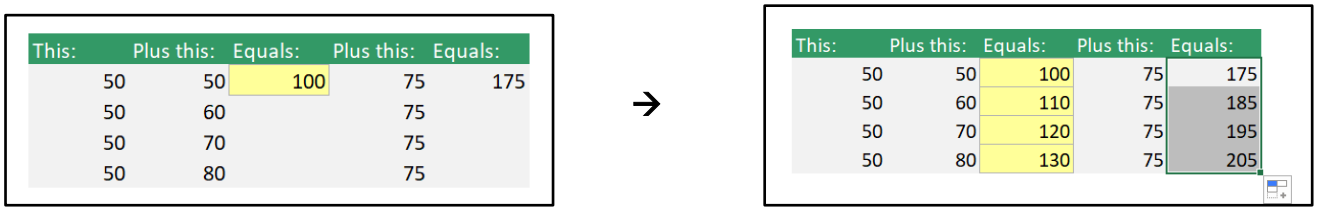
2. **Filling and Autofill**
   - You can quickly fill a series of cells with incremental values using Autofill:

   (a) Enter the first few values in a series

   (b) Select those cells

   (c) Hover over the bottom right corner until you see a small square handle

   (d) Click and drag to fill the series down or across



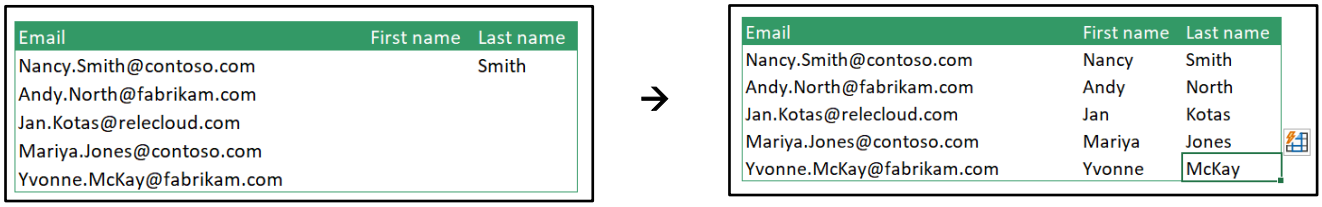
This:	Plus this:	Equals:	Plus this:	Equals:
50	50	100	75	175
50	60		75	
50	70		75	
50	80		75	

→

This:	Plus this:	Equals:	Plus this:	Equals:
50	50	100	75	175
50	60	110	75	185
50	70	120	75	195
50	80	130	75	205

3. Data Splitting

- You can split text into multiple columns based on a delimiter like a comma or space:
  - **(a)** Select the column with the text you want to split
  - **(b)** Go to Data > Text to Columns
  - **(c)** Choose the delimiter and the data format
  - **(d)** Excel will split the text into separate columns



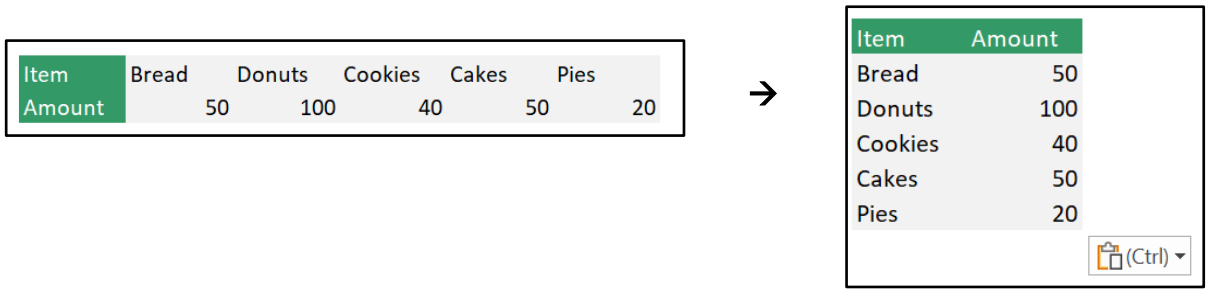
Email	First name	Last name
Nancy.Smith@contoso.com		Smith
Andy.North@fabrikam.com		
Jan.Kotas@relecloud.com		
Mariya.Jones@contoso.com		
Yvonne.McKay@fabrikam.com		

→

Email	First name	Last name
Nancy.Smith@contoso.com	Nancy	Smith
Andy.North@fabrikam.com	Andy	North
Jan.Kotas@relecloud.com	Jan	Kotas
Mariya.Jones@contoso.com	Mariya	Jones
Yvonne.McKay@fabrikam.com	Yvonne	McKay

4. Transposing Data

- To flip rows and columns, you can use the Paste Special feature:
  - **(a)** Copy the cells you want to transpose
  - **(b)** Select the top-left cell where you want the transposed data
  - **(c)** Right-click and choose Paste Special
  - **(d)** Select the Transpose option and click OK



Item	Bread	Donuts	Cookies	Cakes	Pies
Amount	50	100	40	50	20

→

Item	Amount
Bread	50
Donuts	100
Cookies	40
Cakes	50
Pies	20

(Ctrl)

5. Sorting and Filtering Data

- You can sort data alphabetically, numerically, or by format:
  - **(a)** Select the data range
  - **(b)** Go to Data > Sort
  - **(c)** Choose the column to sort by and the sort order
  - **(d)** Filtering lets you show only the rows that meet your criteria:
  - **(e)** Select the data range
  - **(f)** Go to Data > Filter
  - **(g)** Click the filter arrow on the column header
  - **(h)** Select the criteria to filter by.

Department	Category	Oct	Nov	Dec
Meat	Beef	$90,000	$1,10,000	$1,20,000
Bakery	Desserts	$25,000	$80,000	$1,20,000
Produce	Fruit	$10,000	$30,000	$40,000
Produce	Veggies	$30,000	$80,000	$30,000
Deli	Salads	$90,000	$35,000	$25,000
Meat	Chicken	$75,000	$82,000	$20,00,000
Bakery	Breads	$30,000	$15,000	$20,000
Deli	Sandwiches	$80,000	$40,000	$20,000

→

Department	Category	Oct	Nov	Dec
Bakery	Desserts	$25,000	$80,000	$1,20,000
Bakery	Breads	$30,000	$15,000	$20,000
Deli	Salads	$90,000	$35,000	$25,000
Deli	Sandwiches	$80,000	$40,000	$20,000
Meat	Beef	$90,000	$1,10,000	$1,20,000
Meat	Chicken	$75,000	$82,000	$20,00,000
Produce	Fruit	$10,000	$30,000	$40,000
Produce	Veggies	$30,000	$80,000	$30,000

6. Drop Down Lists

- You can create drop-down lists for data validation:
  - **(a)** Select the cells to contain the list
  - **(b)** Go to Data > Data Validation
  - **(c)** Choose List under the Allow dropdown
  - **(d)** Enter the list items separated by commas

Food	Department
Apples	
Beef	
Bananas	
Lemons	
Broccoli	
Kale	
Ham	
Bread	
Chicken	
Cookies	
Cakes	
Pies	

→

Food	Department
Apples	Produce
Beef	Meat
Bananas	Produce
Lemons	
Broccoli	
Kale	
Ham	
Bread	
Chicken	
Cookies	
Cakes	
Pies	

Produce
Meat
Bakery

7. Inserting Charts

- To visualize data with a chart:
  - **(a)** Select the data range to chart
  - **(b)** Go to Insert and choose a chart type
  - **(c)** Excel will insert the chart and you can customize it

8. Pivot Tables

- Pivot tables summarize and analyze large datasets:
  - **(a)** Select the data range
  - **(b)** Go to Insert > Pivot Table
  - **(c)** Drag fields to the Rows, Columns, Values and Filters areas
  - **(d)** Excel will generate a dynamic summary table

Pivot Table 1

Sales	Sep	Oct	Nov	Total
Apples	250	590		840
John		180		180
Mike		120		120
Pete		290		290
Sally	250			250
Bananas		430	600	1030
John			400	400
Mike			200	200
Pete		180		180
Sally		250		250
Cherries	580	910		1490
John		250		250
Mike	250	330		580
Pete		330		330
Sally	330			330
Oranges		120	720	840
John			120	120
Mike			400	400
Pete		120		120
Sally			200	200
Total	830	2050	1320	4200

Pivot Table 2

Month	(All)				
Sales / Reseller	Product / Apples	Bananas	Cherries	Oranges	Total
John	$180	$400	$250	$120	$950
Mike	$120	$200	$580	$400	$1,300
Pete	$290	$180	$330	$120	$920
Sally	$250	$250	$330	$200	$1,030
Total	$840	$1,030	$1,490	$840	$4,200

Pivot Table 3

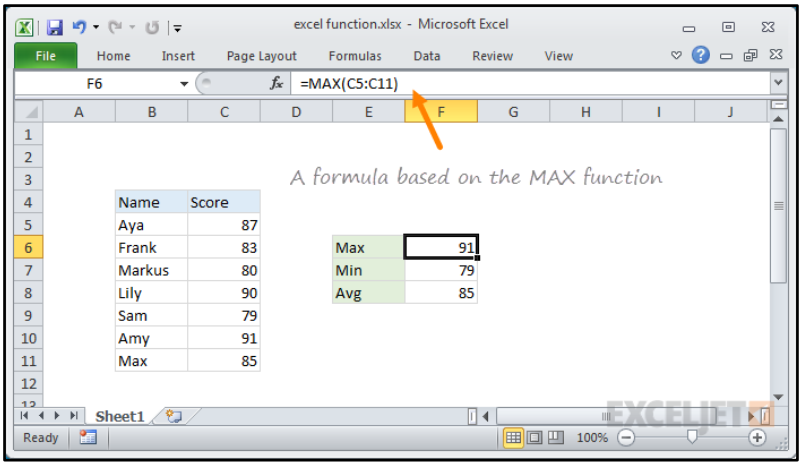
Product	(All)			
Sales / Reseller	Month / Sep	Oct	Nov	Total
John		$430	$520	$950
Mike	$250	$450	$600	$1,300
Pete		$920		$920
Sally	$580	$250	$200	$1,030
Total	$830	$2,050	$1,320	$4,200

9. Formulas

- Formulas perform calculations on values. Some key formulas are:
  (a) Addition: =A1+B1
  (b) Subtraction: =A1-B1
  (c) Multiplication: =A1*B1
  (d) Division: =A1/B1

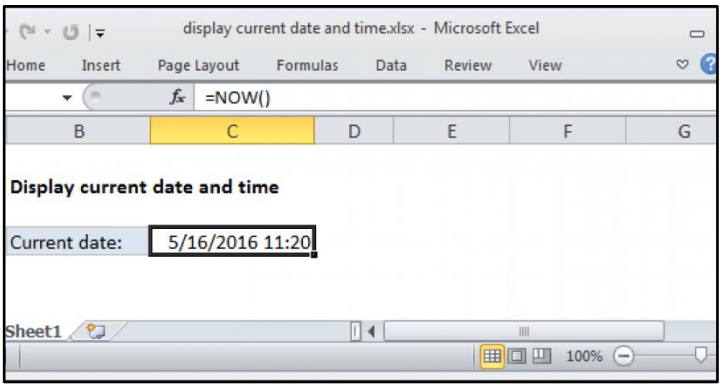
10. Functions

- Functions are predefined formulas that perform specific calculations:
  (a) SUM adds a range of cells: =SUM(A1:A10)
  (b) AVERAGE calculates the average: =AVERAGE(A1:A10)
  (c) MIN returns the minimum value: =MIN(A1:A10)
  (d) MAX returns the maximum value: =MAX(A1:A10)



11. Date and Time

- Excel stores dates and times as serial numbers:
  (a) To add days, add the number of days
  (b) To add hours/minutes, use decimal fractions of a day



12. Text Functions

- Text functions manipulate text strings:
  (a) CONCATENATE joins text: =CONCATENATE(A1,B1)
  (b) LEFT/RIGHT/MID extract substrings

13. IF Statements

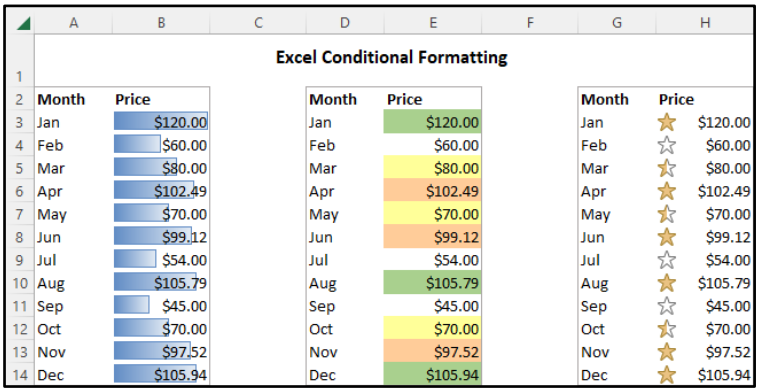
- IF statements perform different actions based on a logical test:
  =IF(A1>0,"Positive","Negative")
  =IF(AND(A1>0,B1>0),"Both Positive","One is Negative")

14. VLOOKUP

- VLOOKUP looks up a value in a table:
  =VLOOKUP(lookup_value,table_array,col_index_num,range_lookup)
- Looks up "apple" in the first column of the table, returns the value in the 2nd column

15. Conditional Formatting

- Conditional formatting highlights cells based on their values:
  (a) Select the cells to format
  (b) Go to Home > Conditional Formatting
  (c) Choose a formatting rule, like "Greater Than"
  (d) Set the criteria and formatting



CHAPTER 6 – POWER BI

Mentor: Ms. Sanvee Khot (Data Science Trainer)

Power BI is a powerful business analytics service provided by Microsoft that allows users to visualize, analyze, and share data insights. It is a collection of software services, apps, and connectors that work together to transform data from various sources into interactive dashboards and reports. With its user-friendly interface, robust data connectivity, and advanced analytics capabilities, Power BI has emerged as a leading business intelligence platform to help organizations make data-driven decisions.

Key features of Power BI include:

1. Connecting to and integrating data from multiple sources like Excel, SQL Server, Azure, Salesforce, etc.
2. Creating customizable dashboards and reports with rich visualizations and AI-powered insights
3. Enabling collaboration by allowing users to share reports and dashboards
4. Providing mobile apps for iOS, Android and Windows to access data on-the-go
5. Offering an on-premises reporting server called Power BI Report Server

The core components of Power BI are:

1. **Power BI Desktop -** A free desktop application for connecting to data, shaping and modeling the data, and creating reports.
2. **Power BI Service -** The cloud-based platform for publishing, sharing, and collaborating on reports.
3. **Power BI Mobile Apps -** Apps for iOS, Android, and Windows that allow you to view reports on the go.

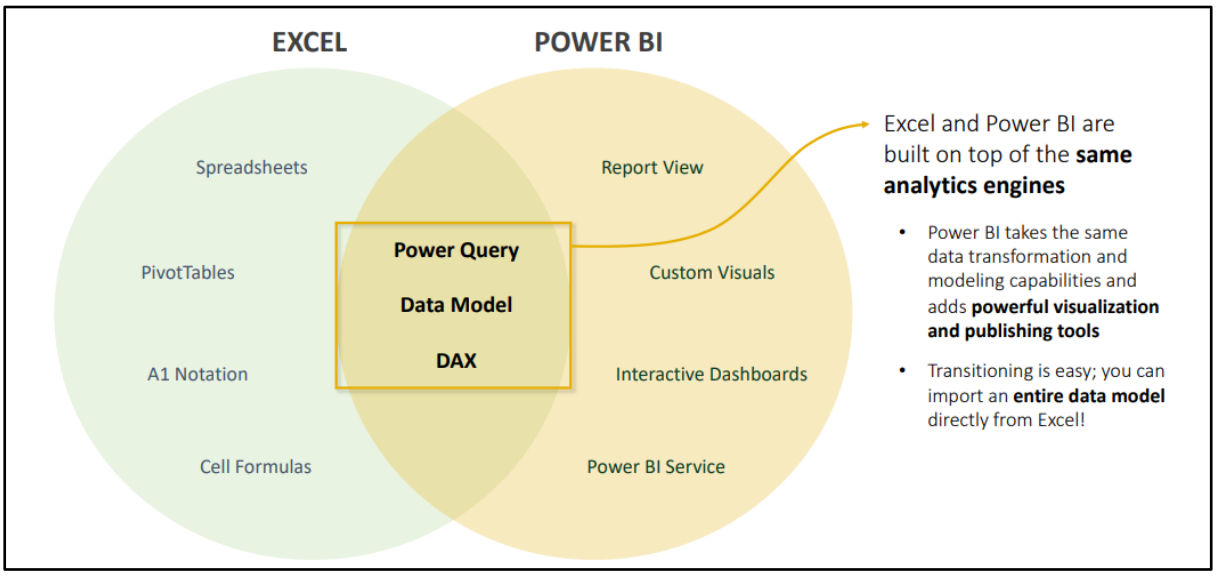
Power BI is available in different **licensing plans** to suit various business needs:

1. Power BI Pro for individual users to create and publish reports
2. Power BI Premium per user and per capacity for enterprise-wide deployments
3. Embedded analytics for developers to embed Power BI into their applications

Power BI enables users to connect to a wide range of data sources, transform the data using powerful data preparation tools, build interactive visualizations and dashboards, and share insights across the organization.

How is Power BI different from MS-EXCEL ?

Feature	Excel	Power BI
Data Handling	Better for smaller datasets	Handles large datasets efficiently
	May struggle with performance for large data	Efficiently handles millions of rows
Data Visualization	Static charts and graphs	Dynamic, interactive visualizations and dashboards
	Limited set of visualization options	Wide range of built-in and custom visuals
Collaboration & Sharing	Sharing via email or online tools	Cloud-based sharing of reports and dashboards
	More challenging for collaboration	Easier for team collaboration
Data Modeling	Basic data modeling features	Advanced data modeling capabilities
	Less scalable for complex datasets	Manages complex, interconnected datasets
Formulas & Functions	Rich library of built-in functions	DAX formulas and measures



Learning about the Dataset and Handling the Dataset in Power BI

The first step in working with Power BI is to understand the dataset you will be using. Examine the data sources, the structure of the data, and the relationships between different tables or entities. This will help you plan how to model the data effectively in Power BI.

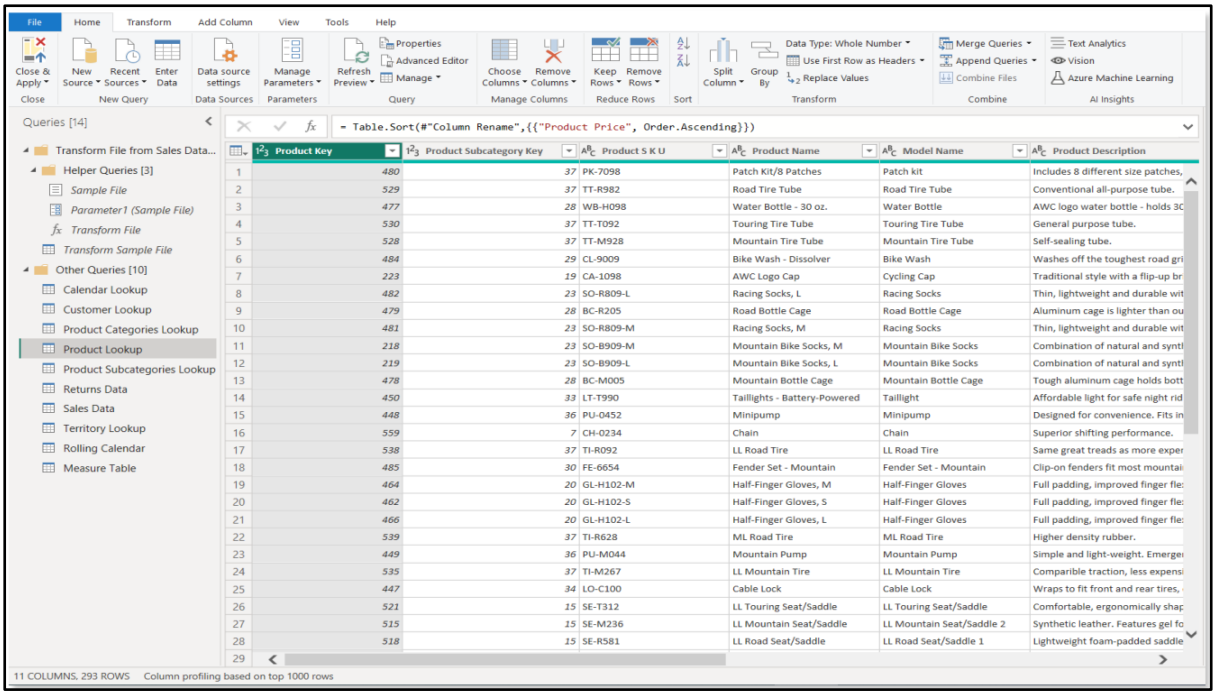
Once you have a good understanding of the dataset, you can start importing the data into Power BI Desktop. Power BI supports connecting to a variety of data sources, including Excel, databases, cloud services, and big data sources.

When the data is imported, you can review the data types, check for any errors or missing values, and perform initial data cleaning and transformation as needed.

Manipulating and Editing Data using Power Query

Power Query is the data preparation and transformation engine in Power BI Desktop. It allows you to perform a wide range of data manipulation tasks, such as:

1. Merging and appending data from multiple sources
2. Cleaning and formatting data (e.g. removing duplicates, handling null values)
3. Applying custom transformations and calculations
4. Unpivoting and pivoting data
5. Grouping and aggregating data



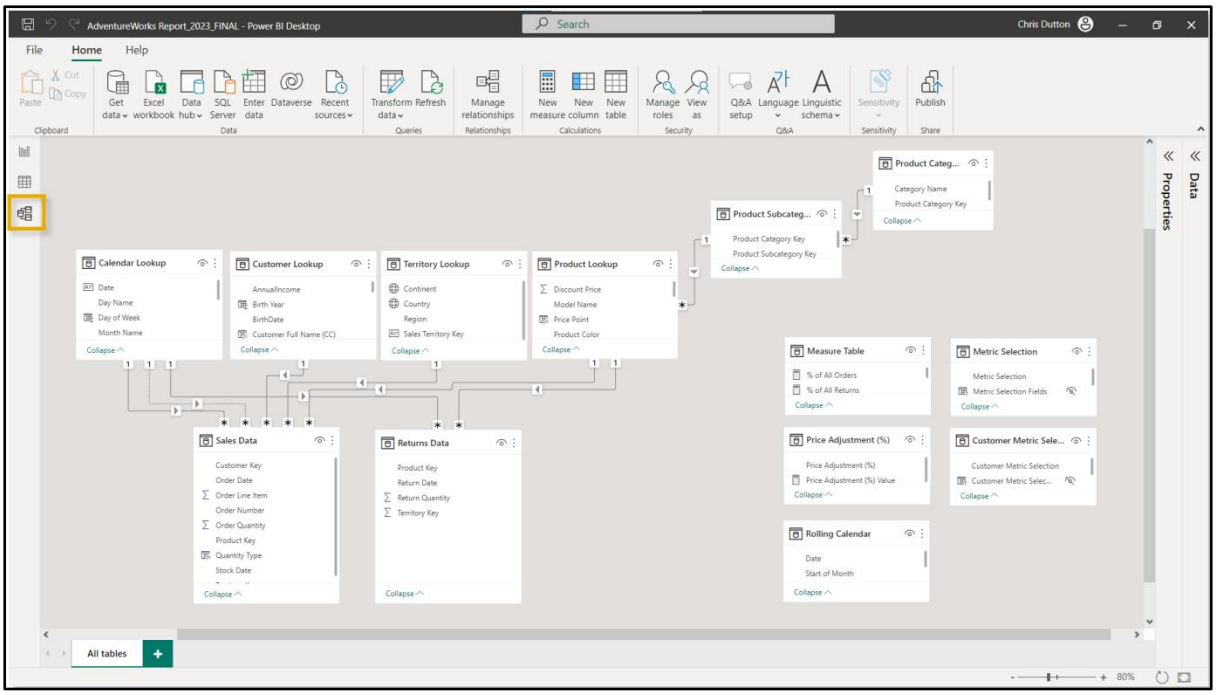
By using Power Query, you can shape the data into a format that is optimized for analysis and reporting in Power BI.

Model View of the Dataset

The Model view in Power BI Desktop is where you define the data model - the relationships between tables, create calculated columns, and define measures.

In the Model view, you can:

1. Create relationships between tables based on common columns
2. Manage data types and formats
3. Define calculated columns using DAX (Data Analysis Expressions) formulas
4. Create measures, which are calculated fields that can be used in visualizations



Designing an effective data model is crucial for enabling complex analysis and creating meaningful visualizations in Power BI.

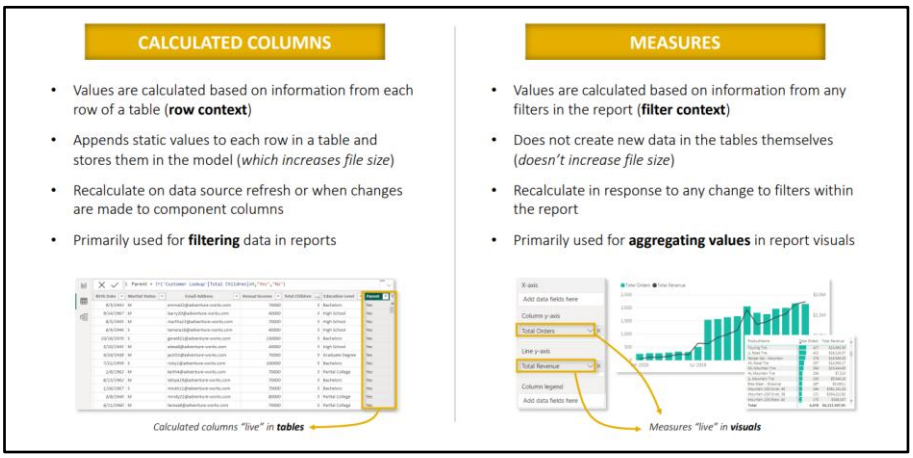
Calculating Measures using DAX Queries

DAX (Data Analysis Expressions) is the formula language used in Power BI to create calculated measures and columns. Measures are calculated fields that can be used in visualizations to provide insights and analysis.

Some common examples of measures include:

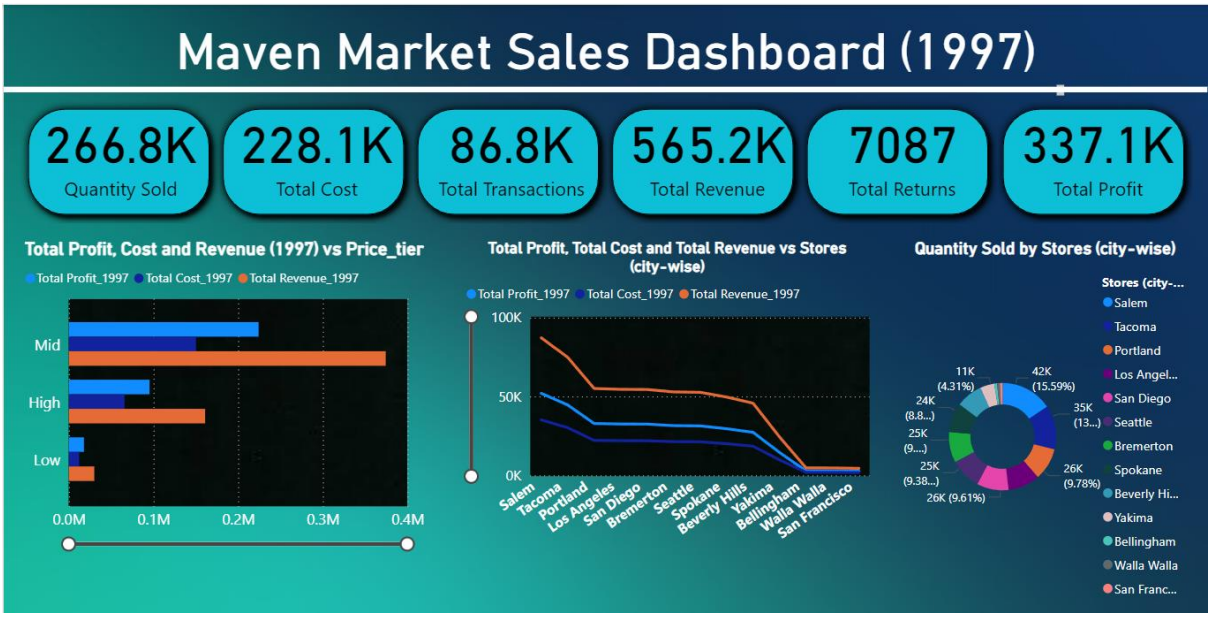
1. Total Sales
2. Average Order Value
3. Year-over-Year Growth
4. Customer Churn Rate

By creating custom measures using DAX, you can unlock deeper insights from your data and enable more sophisticated reporting and analysis.



Visualization of Dataset by designing a Dashboard

The final step in the Power BI workflow is to create interactive visualizations and dashboards to present the data insights. Power BI offers a wide range of visualization types, including charts, graphs, maps, cards, and custom visuals.



In the Report view, you can:

1. Add visualizations to the report canvas
2. Configure the data sources and fields for each visualization
3. Customize the appearance and formatting of the visualizations
4. Create interactive filters and slicers to allow users to explore the data

Once the report is designed, you can publish it to the Power BI Service, where it can be shared with other users, embedded in applications, or accessed on mobile devices.

By following these steps, you can leverage the full capabilities of Power BI to transform raw data into meaningful, actionable insights that drive better business decisions.

CHAPTER 7 – SKILLS ACQUIRED

Python and Python Libraries

1. **Python Fundamentals:** Understanding the basic syntax, data types, control structures, and functions in Python.
2. **pandas:** Proficiency in using the pandas library for data manipulation, cleaning, and analysis. This includes working with DataFrames, Series, indexing, filtering, grouping, and applying various pandas functions.
3. **NumPy:** Expertise in using the NumPy library for working with multi-dimensional arrays, performing mathematical operations, and leveraging its powerful functions for data analysis.
4. **Matplotlib:** Ability to create high-quality, publication-ready visualizations using the Matplotlib library, including line plots, scatter plots, histograms, bar charts, and more.
5. **Data Cleaning and Preprocessing:** Skills in handling missing data, removing duplicates, handling outliers, and transforming data into a format suitable for analysis.
6. **Data Exploration and Analysis:** Competence in exploring datasets, identifying patterns, and deriving insights using Python's data analysis capabilities.

SQL (Structured Query Language)

1. **SQL Fundamentals:** Understanding SQL syntax, including SELECT, FROM, WHERE, JOIN, GROUP BY, and ORDER BY clauses, as well as SQL data types and functions.
2. **SQL Queries:** Ability to write complex SQL queries to extract, filter, and aggregate data from relational databases.
3. **SQL Joins:** Proficiency in performing different types of SQL joins (e.g., inner, outer, left, right) to combine data from multiple tables.

Excel and Power BI

1. **Advanced Excel:** Expertise in using Excel's advanced features, such as pivot tables, lookups, data validation, and macros, to perform data analysis and reporting.
2. **Power Query:** Ability to use Power Query (in Excel or Power BI) to extract, transform, and load data from various sources, including web pages, databases, and text files.
3. **Power BI:** Competence in using Power BI to create interactive, visually appealing dashboards and reports that provide insights and support decision-making.
4. **Data Visualization:** Skills in creating effective and informative data visualizations using tools like Matplotlib, Seaborn, Plotly, and Power BI's visualization capabilities.
5. **Presenting Data:** Ability to present data insights in a clear, concise, and compelling manner to stakeholders and decision-makers.
6. **Problem-Solving and Critical Thinking:** Developing the ability to identify and define problems, gather and analyze relevant data, and devise effective solutions.

By mastering these skills, one will be well-equipped to tackle a wide range of data analytics tasks, from data extraction and transformation to exploratory data analysis, visualization, and reporting, using a combination of Python, SQL, Excel, and Power BI.

CHAPTER 8 – CONCLUSION

Data Analytics Training Summary

Skills Learned:

1. **Python and its Libraries (pandas, NumPy, Matplotlib):** Data manipulation, cleaning, analysis, and visualization.
2. **SQL:** Extracting, filtering, and aggregating data from databases.
3. **Advanced Excel:** Utilizing pivot tables, lookups, and macros for analysis and reporting.
4. **Power BI:** Creating interactive dashboards and reports with actionable insights.

Outcomes:

1. Extract, clean, and transform data.
2. Perform exploratory data analysis.
3. Create compelling data visualizations.
4. Communicate data-driven insights.

Benefits:

1. Strong foundation for tackling various data analytics challenges.
2. Enhanced critical thinking and problem-solving skills.
3. Ability to drive innovation, improve business performance, and make impactful contributions.

The training in data analytics, covering Python and its libraries (pandas, NumPy, Matplotlib), SQL, advanced Excel, and Power BI, has equipped me with a comprehensive set of skills necessary for effective data analysis and decision-making.

By combining the skills acquired in Python, SQL, Excel, and Power BI, I am now well-prepared to tackle a wide range of data analytics challenges. I can efficiently extract, clean, and transform data, perform exploratory data analysis, create compelling visualizations, and communicate their findings to stakeholders effectively.

The training has not only imparted technical skills but also fostered critical thinking and problem-solving abilities, which are essential for making data-driven decisions. As I embark on their data analytics journeys, they can confidently apply the knowledge gained during the training to drive innovation, improve business performance, and make a meaningful impact in their respective fields.

CHAPTER 9 – REFERENCES

1. Simplilearn. (2021, April 16). Data Analyst: Job Description, Responsibilities and Skills Required. Simplilearn.com. https://www.simplilearn.com/data-analyst-job-description-article
2. Data Analyst vs. Data Scientist: What's the Difference? (2023, November 29). Coursera. https://www.coursera.org/in/articles/data-analyst-vs-data-scientist-whats-the-difference
3. The Types of Data Science Roles Explained. (2022, October 7). 365 Data Science. https://365datascience.com/career-advice/types-of-data-science-roles-explained/
4. Python Software Foundation. (n.d.). What Is Python? Executive Summary. Python. https://www.python.org/doc/essays/blurb/
5. What is Python? (2023, June 4). GeeksforGeeks. https://www.geeksforgeeks.org/what-is-python/
6. VanderPlas, J. (2019). Python Data Science Handbook | Python Data Science Handbook. Github.io. https://jakevdp.github.io/PythonDataScienceHandbook/
7. Python. (2020). The Python Standard Library — Python 3.8.1 documentation. Python.org. https://docs.python.org/3/library/
8. *SQL Tutorial*. (n.d.). https://www.w3schools.com/sql/
9. Groff, J., Weinberg, P., & Oppel, A. (2009, August 12). *SQL The Complete Reference, 3rd Edition*. Mcgraw-hill.
10. *Learn MySQL: A Comprehensive Guide to Using and Managing Databases*. (n.d.). https://www.w3schools.in/mysql/
11. *SQL Tutorial, Tutorials SQL*. (n.d.). https://beginner-sql-tutorial.com/sql.htm
12. Introduction to MS-Excel. (2021, May 29). GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-ms-excel/
13. What is MS Excel? | An Overview, Features & History. (n.d.). BYJUS. https://byjus.com/govt-exams/ms-excel-basics/
14. Create a simple formula in Excel. (n.d.). Support.microsoft.com. https://support.microsoft.com/en-us/office/create-a-simple-formula-in-excel-11a5f0e5-38a3-4115-85bc-f4a465f64a8a
15. The 15 Basic Excel Formulas Everyone Needs to Know. (2023, December) https://www.datacamp.com/tutorial/basic-excel-formulas-for-everyone
16. Power BI Tutorial - Tutorialspoint. (n.d.). Www.tutorialspoint.com. https://www.tutorialspoint.com/power_bi/index.htm
17. Microsoft. (2023, February 22). What Is Power BI? - Power BI. Learn.microsoft.com. https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview
18. Microsoft. (2024). Power BI - Data Visualization | Microsoft Power Platform. Www.microsoft.com. https://www.microsoft.com/en-us/power-platform/products/power-bi
19. Power BI vs. Excel: Three Major Differences. (n.d.). Intellipaat.com. https://intellipaat.com/blog/power-bi-vs-excel/