

SQL: Capstone Project

Session 2: Creating a Database

1.
 - a. Create a database with the name: AbleJobs
 - b. Create the following Table with the name: Sales1
 - c. Display all the data in the above table

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, which includes the 'ablejobs' schema selected. The main pane shows a SQL editor with the following code:

```
1  # Session 2 : Creating a Database
2  #2.1.a -Create a database with the name: AbleJobs
3  CREATE DATABASE Able_Jobs;
4  USE Able_Jobs;
5  # 2.1.b - Create the Table with the name: Sales1
6  CREATE TABLE Sales1 (
7      salesman_id INT PRIMARY KEY,
8      name VARCHAR (30),
9      city VARCHAR (30),
10     commission FLOAT
11 );
12 INSERT INTO Sales1
13 VALUES (5001, 'James Hoog', 'New York', '0.15'),
14     (5002, 'Nail Knite', 'Paris', '0.13'),
15     (5005, 'Pit Alex', 'London', '0.11'),
16     (5006, 'Mc Lyon', 'Paris', '0.14'),
17     (5007, 'Paul Adam', 'Rome', '0.13'),
18     (5003, 'Lauson Hen', 'San Jose', '0.12');
19 # 2.1.c- Display all the data in the above table
20 SELECT * FROM Sales1;
```

The 'Result Grid' tab shows the data from the Sales1 table:

| salesman_id | name | city | commission |
|-------------|------------|----------|------------|
| 5001 | James Hoog | New York | 0.15 |
| 5002 | Nail Knite | Paris | 0.13 |
| 5003 | Lauson Hen | San Jose | 0.12 |
| 5005 | Pit Alex | London | 0.11 |
| 5006 | Mc Lyon | Paris | 0.14 |
| 5007 | Paul Adam | Rome | 0.13 |
| * | NULL | NULL | NULL |

The 'Output' tab shows the message: "27 16:22:57 INSERT INTO Sales1 VALUES (5001, 'James Hoog', 'New York', '0.15'), (5002, 'Nail Knite', 'Paris', '0.13')... 6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0".

2.
 - a. Create a database with the name: AbleJobs
 - b. Create the following Table with the name: Sales
 - c. Display all the data in the above table

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: oyee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hiv* SQL File 29 SQL File 30 SQL_miniproject Session2 SQL File 33* SQL File 34* SQL File 35

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - Views
 - Stored Procedures
 - Functions
- ablejobs
- ibm_comer
- miniproject_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- moviesdb
- mydb
- sakila
- sys
- world

Administration Schemas

Information

Schema: ablejobs

```

22 #2.1.a - Create a database with the name: AbleJobs
23 • CREATE DATABASE Able_Jobs;
24 • USE Able_Jobs;
25 #2.1.b - Create the Table with the name: Sales2
26 • CREATE TABLE Sales2 (
27     customer_id INT PRIMARY KEY,
28     cust_name VARCHAR (30),
29     city VARCHAR (30),
30     grade INT,
31     salesman_id INT
32 );
33 • INSERT INTO Sales2
34     VALUES (3002, 'Nick Rimando', 'New York', 100, 5001),
35     (3007, 'Brad Davis', 'New York', 200, 5001),
36     (3005, 'Graham Zusi', 'California', 200, 5002),
37     (3008, 'Julian Green', 'London', 300, 5002),
38     (3004, 'Fabian Johnson', 'Paris', 300, 5006),
39     (3009, 'Geoff Cameron', 'Berlin', 100, 5003),
40     (3003, 'Jozy Altidore', 'Moscow', 200, 5007);
41 # 2.1.c - Display all the data in the above table
42 • SELECT * FROM Sales2;

```

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

| customer_id | cust_name | city | grade | salesman_id |
|-------------|----------------|------------|-------|-------------|
| 3002 | Nick Rimando | New York | 100 | 5001 |
| 3003 | Jozy Altidore | Moscow | 200 | 5007 |
| 3004 | Fabian Johnson | Paris | 300 | 5006 |
| 3005 | Graham Zusi | California | 200 | 5002 |
| 3007 | Brad Davis | New York | 200 | 5001 |
| 3008 | Julian Green | London | 300 | 5002 |
| 3009 | Geoff Cameron | Berlin | 100 | 5003 |
| 3001 | Walter | India | 200 | 5008 |

Sales2 6 | Output | Action Output | Object Info | Session | Message | Duration / Fetch |

Object Info Session # Time Action 30 16:24:05 INSERT INTO Sales2 VALUES (3002, 'Nick Rimando', 'New York', 100, 5001), (3007, 'Brad Davis', 'New York', 200, 5001), (3005, 'Graham Zusi', 'California', 200, 5002), (3008, 'Julian Green', 'London', 300, 5002), (3004, 'Fabian Johnson', 'Paris', 300, 5006), (3009, 'Geoff Cameron', 'Berlin', 100, 5003), (3003, 'Jozy Altidore', 'Moscow', 200, 5007); 0.016 sec

Query Completed

Session 3: Other Basic Queries

- Create a database with the name: AbleJobs
- Create the following Table with the name: Sales1
- In the above table, write a SQL query to change the following data:
 - Change commission of salesman with name of 'Pit Alex' to 0.22
 - Change city of salesman with salesman_id of '5003' to Paris
- Display all the data in the above table

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: oyee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hiv* SQL File 29 SQL File 30 SQL_miniproject Session2 SQL File 33* SQL File 34* SQL File 35

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - Views
 - Stored Procedures
 - Functions
- ablejobs
- ibm_comer
- miniproject_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- moviesdb
- mydb
- sakila
- sys
- world

Administration Schemas

Information

Schema: ablejobs

```

2 #3.1.a - Create a database with the name: AbleJobs
3 • USE Able_Jobs;
4 #3.1.b - Create the Table with the name: Sales1
5 • CREATE TABLE Sales1 (
6     salesman_id INT PRIMARY KEY,
7     name VARCHAR (30),
8     city VARCHAR (30),
9     commission DECIMAL (4,2)
10 • INSERT INTO Sales1
11     VALUES (5001, 'James Hoog', 'New York', 0.15),
12     (5002, 'Nail Knite', 'Paris', 0.13),
13     (5003, 'Lawnn Hem', 'Paris', 0.12),
14     (5004, 'Pit Alex', 'London', 0.12),
15     (5005, 'Mc Lyon', 'Paris', 0.14),
16     (5006, 'Paul Adam', 'Rome', 0.13),
17     (5007, 'Paul Adams', 'Rome', 0.13);
18 #3.1.c - In the Sales1 table, write a SQL query to change the following data:
19 # i - Change commission of salesman with name of 'Pit Alex' to 0.22
20 • UPDATE Sales1
21     SET commission = '0.22';
22 WHERE name = 'Pit Alex';
23 #ii- Change city of salesman with salesman_id of '5003' to Paris
24 • UPDATE Sales1
25     SET city = 'Paris';
26 WHERE salesman_id = '5003';
27 #3.1.d - Display all the data
28 • SELECT * FROM Sales1;

```

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

| salesman_id | name | city | commission |
|-------------|------------|----------|------------|
| 5001 | James Hoog | New York | 0.15 |
| 5002 | Nail Knite | Paris | 0.13 |
| 5003 | Lawnn Hem | Paris | 0.12 |
| 5004 | Pit Alex | London | 0.12 |
| 5005 | Mc Lyon | Paris | 0.14 |
| 5006 | Paul Adam | Rome | 0.13 |
| 5007 | Paul Adams | Rome | 0.13 |
| 5008 | Walter | India | 0.12 |

Sales1 5 | Output | Action Output | Object Info | Session | Message | Duration / Fetch |

Object Info Session # Time Action 33 16:25:44 UPDATE Sales1 SET city = 'Paris' WHERE salesman_id = '5003' 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 0.000 sec

Query Completed

- 2.
- Create a database with the name: AbleJobs
 - Create the following Table with the name: Sales2
 - In the above table, write a SQL query to alter the following data:
 - Change grade of customer with name of 'Graham Zusi' to 300
 - Change city of customer with cust_id of '3009' to London
 - Display all the data in the above table

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance sql course Local instance MySQL80
Navigator
SCHEMAS
ablejobs
able_homeworks
able_jobs
Tables
Views
Stored Procedures
Functions
cust_omer
miniproject_db
Tables
Views
Stored Procedures
Functions
moviesdb
mydb
sakila
sys
world
Administration Schemas
Information
Schema: ablejobs

17 #3.2.a - Create a database with the name: AbleJobs
18 • USE Able_jobs;
19 #3.2.b- Create the Table with the name: Sales2
20 • SELECT * FROM Sales2;
21 #3.2.c - In the Sales2 table, write a SQL query to alter the following data:
22     #i-Change grade of customer with name of 'Graham Zusi' to 300
23 • UPDATE Sales2
24     SET grade = '300'
25     WHERE cust_name = 'Graham Zusi';
26     #ii- Change city of customer with cust_id of '3009' to London
27 • UPDATE Sales2
28     SET city = 'London'
29     WHERE customer_id = '3009';
30
31 #3.2.e - Display all the data
32 • SELECT * FROM Sales2;

```

| customer_id | cust_name | city | grade | salesman_id |
|-------------|----------------|------------|-------|-------------|
| 3002 | Nick Rimando | New York | 100 | 5001 |
| 3003 | Jozy Altidor | Moscow | 200 | 5007 |
| 3004 | Fabian Johnson | Paris | 300 | 5006 |
| 3005 | Graham Zusi | California | 300 | 5002 |
| 3007 | Brad Davis | New York | 200 | 5001 |
| 3008 | Julian Green | London | 300 | 5002 |
| 3009 | Geoff Cameron | London | 100 | 5003 |
| NULL | NULL | NULL | NULL | NULL |

Sales2 x

Action Output

Object Info Session

Query Completed

36 16:34:33 UPDATE Sales2 SET city = 'London' WHERE customer_id = '3009'

Message: 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0 Duration / Fetch: 0.016 sec

Session 4: Functions and Wildcards

- Create a database with the name: AbleJobs
- Create the following Table with the name: Sales1
- From the above table, write a SQL query to find the details of those salespeople who come from the 'Paris' City or 'Rome' City. Return salesman_id, name, city, commission.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas

employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30 SQL_miniproject Session2 Session3 SQL File 34* SQL File 35

```

1  # Session 4: Functions and Wildcards
2  #4.1.a - Create a database with the name: AbleJobs
3  USE Able_jobs;
4  #4.1.b - Create the following Table with the name: Sales1
5  SELECT * FROM Sales1;
6
7  /* 4.1.c - From the Sales1 table, write a SQL query to find the details of those salespeople who come from the 'Paris' City
8  or 'Rome' City. Return salesman_id, name, city, commission. */
9  SELECT salesman_id, name, city, commission FROM Sales1
10 WHERE city = 'Paris' OR city = 'Rome';
11
12

```

Result Grid

| salesman_id | name | city | commission |
|-------------|------------|-------|------------|
| 5002 | Nial Krite | Paris | 0.13 |
| 5003 | Lauzon Hen | Paris | 0.12 |
| 5006 | McLyon | Paris | 0.14 |
| 5007 | Paul Adam | Rome | 0.13 |
| 5008 | | | |

Sales1 7 x

Action Output

| # | Time | Action |
|----|----------|---|
| 43 | 16:40:55 | SELECT * FROM Sales1 WHERE city = 'Paris' OR city = 'Rome' LIMIT 0, 10000 |

Object Info Session Query Completed

- d. From the following table, write a SQL query to find the details of those salespeople who live in cities apart from 'Paris' and 'Rome'. Return salesman_id, name, city, commission.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas

employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30 SQL_miniproject Session2 Session3 SQL File 34* SQL File 35

```

10 • SELECT salesman_id, name, city, commission
11   FROM Sales1
12   WHERE city = 'Paris' OR city = 'Rome';
13
14 /* 4.1.d - From the Sales1 table, write a SQL query to find the details of those salespeople who live in cities apart from
15 'Paris' and 'Rome'. Return salesman_id, name, city, commission.*/
16
17 • SELECT salesman_id, name, city, commission
18   FROM Sales1
19   WHERE city != 'Paris' AND city != 'Rome';
20
21
22
23
24

```

Result Grid

| salesman_id | name | city | commission |
|-------------|------------|----------|------------|
| 5001 | James Hoog | New York | 0.15 |
| 5005 | Pit Alex | London | 0.22 |
| 5008 | | | |

Sales1 10 x

Action Output

| # | Time | Action |
|----|----------|-------------------------------------|
| 46 | 16:45:44 | SELECT * FROM Sales1 LIMIT 0, 10000 |

Object Info Session Query Completed

- e. From the following table, write a SQL query to find the details of salespeople who get the commission in the range from 0.12 to 0.14 (begin and end values are included). Return salesman_id, name, city, and commission.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator: schemas employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34* Limit to 10000 rows
Schemas
No object selected
Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: 
salesman_id name city commission
5002 Nall Krite Paris 0.13
5003 Lauson Hen Paris 0.12
5006 Mc Lyon Paris 0.14
5007 Paul Adam Rome 0.13
NULL NULL NULL NULL

```

Sales1 39 x

Action Output

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|-------------------|-----------------------|
| 99 | 17:48:41 | SELECT customer_id, cust_name, city, grade, salesman_id FROM Sales2 WHERE cust_name LIKE "%n%" | 3 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Query Completed

- f. From the following table, write a SQL query to find the details of those salespeople whose name starts with any letter within 'A' and 'L' (not inclusive). Return salesman_id, name, city, commission.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator: schemas employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34* Limit to 10000 rows
Schemas
No object selected
Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: 
salesman_id name city commission
5001 James Hoog New York 0.15
NULL NULL NULL NULL

```

Sales1 40 x

Action Output

| # | Time | Action | Message | Duration / Fetch |
|-----|----------|--|-------------------|-----------------------|
| 100 | 17:52:45 | SELECT salesman_id, name, city, commission FROM Sales1 WHERE commission >= '0.119' AND commis... | 4 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Query Completed

- g. From the following table, write a SQL query to find the details of the customers whose name begins with the letter 'B'. Return customer_id, cust_name, city, grade, salesman_id.

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree is visible, showing databases like 'analysts', 'employee_table', 'ibm employees - sheet1', 'ibm_emp', and others. The main area contains a SQL editor window with the following code:

```

37 /* 4.1.g - From the Sales2 table, write a SQL query to find the details of the customers
38   whose name begins with the letter 'B'. Return customer_id, cust_name, city, grade, salesman_id. */
39
40 • SELECT customer_id, cust_name, city, grade, salesman_id
41   FROM Sales2
42   WHERE cust_name LIKE 'B%';
43
44
45
46
47
48
49
50
51
52
53

```

Below the SQL editor is a 'Result Grid' showing the query results:

| customer_id | cust_name | city | grade | salesman_id |
|-------------|------------|----------|-------|-------------|
| 3007 | Brad Davis | New York | 200 | 5001 |
| NULL | NULL | NULL | NULL | NULL |

At the bottom, the 'Output' pane displays the execution message and duration:

Sales2 41 x
Output
Action Output # Time Action Message Duration / Fetch
Object Info Session 101 17:53:24 SELECT salesman_id, name, city, commission FROM Sales1 WHERE name BETWEEN 'A%' and 'L%' LIM... 1 row(s) returned 0.000 sec / 0.000 sec
Query Completed

- h. From the following table, write a SQL query to find the details of the customers whose names end with the letter 'n'. Return customer_id, cust_name, city, grade, salesman_id.

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree is visible, showing databases like 'analysts', 'employee_table', 'ibm employees - sheet1', 'ibm_emp', and others. The main area contains a SQL editor window with the following code:

```

44
45 /* 4.1.h - From the Sales2 table, write a SQL query to find the details of the customers whose names end with the
46   letter 'n'. Return customer_id, cust_name, city, grade, salesman_id.*/
47
48 • SELECT customer_id, cust_name, city, grade, salesman_id
49   FROM Sales2
50   WHERE cust_name LIKE '%n';
51
52
53
54
55
56
57
58
59
60

```

Below the SQL editor is a 'Result Grid' showing the query results:

| customer_id | cust_name | city | grade | salesman_id |
|-------------|----------------|--------|-------|-------------|
| 3004 | Fabian Johnson | Paris | 300 | 5006 |
| 3008 | Julian Green | London | 300 | 5002 |
| 3009 | Geoff Cameron | London | 100 | 5003 |
| NULL | NULL | NULL | NULL | NULL |

At the bottom, the 'Output' pane displays the execution message and duration:

Sales2 42 x
Output
Action Output # Time Action Message Duration / Fetch
Object Info Session 102 17:54:03 SELECT customer_id, cust_name, city, grade, salesman_id FROM Sales2 WHERE cust_name LIKE 'B%' ... 1 row(s) returned 0.000 sec / 0.000 sec
Query Completed

- i. From the following table, write a SQL query to find the details of those salespeople whose name starts with 'N' and the fourth character is 'I'. Rests may be any character. Return salesman_id, name, city, commission.

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree shows the 'able_jobs' schema with tables like 'sales1', 'sales2', and 'Sales2'. The main query editor window contains the following SQL code:

```

49 FROM Sales2
50 WHERE cust_name LIKE '%n';
51
52
53 /* 4.1.i - From the Sales1 table, write a SQL query to find the details of those salespeople whose name starts with 'N' and the fourth character is 'I'. Rests may be any character. Return salesman_id, name, city, commission.*/
54
55
56 • SELECT salesman_id, name, city, commission
57 FROM Sales1
58 WHERE name LIKE 'N_I%' ;
59
60
61
62
63
64
65

```

The 'Result Grid' below the editor shows the output of the query:

| salesman_id | name | city | commission |
|-------------|------------|-------|------------|
| 5002 | Nail Krite | Paris | 0.13 |
| * | | | |

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

```

Object Info Session
# 113 17:59:26 SELECT salesman_id, name, city, commission FROM Sales1 WHERE name LIKE 'N_I%' LIMIT 0, 10000 1 row(s) returned
Duration / Fetch
0.000 sec / 0.000 sec
Query Completed

```

2.

- a. Create a database with the name: AbleJobs
 b. Create the following Table with the name: Nobel

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree shows the 'able_jobs' schema with tables like 'sales1', 'sales2', and 'Sales2'. The main query editor window contains the following SQL code:

```

60 # 4.2.a - Create a database with the name: AbleJobs
61 USE Able_jobs;
62
63
64 # 4.2.b - Create the following Table with the name: Nobel
65 • CREATE TABLE Nobel1 (
66     YEAR INT,
67     SUBJECT VARCHAR (30),
68     WINNER VARCHAR (50),
69     COUNTRY VARCHAR (30),
70     CATEGORY VARCHAR (40)
71 );
72 • INSERT INTO Nobel(YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY)
73 VALUES
74 (1970, 'Physics', 'Hannes Alfven', 'Sweden', 'Scientist'),
75 (1970, 'Physics', 'Louis Neel', 'France', 'Scientist'),
76 (1970, 'Chemistry', 'Luis Federico Leloir', 'France', 'Scientist'),
77 (1970, 'Literature', 'Joseph Brodsky', 'Russia', 'Literature'),
78 (1970, 'Economics', 'Robert Solow', 'USA', 'Economics')

```

The 'Result Grid' below the editor shows the output of the query:

| YEAR | SUBJECT | WINNER |
|------|------------|------------------------|
| 1987 | Chemistry | Donald J. Cram |
| 1987 | Chemistry | Jean-Marie Lehn |
| 1987 | Physiology | Susumu Tonegawa |
| 1987 | Physics | Johannes Georg Bednorz |
| 1987 | Literature | Joseph Brodsky |
| 1987 | Economics | Robert Solow |

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

```

Object Info Session
# 119 18:37:07 SELECT YEAR, SUBJECT, WINNER FROM NOBEL WHERE YEAR = '1970' LIMIT 0, 10000
Duration / Fetch
0.000 sec / 0.000 sec
Query Completed

```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance sql course Local instance MySQL80

SCHEMAS: analysts employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

Analysts employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

Limit to 10000 rows

Result Grid: Filter Rows: Export: Wrap Cell Content: 15

YEAR SUBJECT WINNER

| YEAR | SUBJECT | WINNER |
|------|------------|------------------------|
| 1987 | Chemistry | Donald J. Cram |
| 1987 | Chemistry | Jean-Marie Lehn |
| 1987 | Physiology | Susumu Tonegawa |
| 1987 | Physics | Johannes Georg Bednorz |
| 1987 | Literature | Joseph Brodsky |
| 1987 | Economics | Robert Solow |

NOBEL 53 x

Output: Action Output

Object Info Session # Time Action Message Duration / Fetch

119 18:37:07 SELECT YEAR, SUBJECT, WINNER FROM NOBEL WHERE YEAR = '1987' LIMIT 0, 10000 8 row(s) returned 0.000 sec / 0.000 sec

Query Completed

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance sql course Local instance MySQL80

SCHEMAS: analysts employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

Analysts employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

Limit to 10000 rows

Result Grid: Filter Rows: Export: Wrap Cell Content: 15

YEAR SUBJECT WINNER

| YEAR | SUBJECT | WINNER |
|------|------------|------------------------|
| 1987 | Physics | Johannes Georg Bednorz |
| 1987 | Literature | Joseph Brodsky |
| 1987 | Economics | Robert Solow |
| 1994 | Literature | Kenzaburo Oe |

100

94 (1987, 'Physics', 'Johannes Georg Bednorz', 'Germany', 'Scientist'),
95 (1987, 'Literature', 'Joseph Brodsky', 'Russia', 'Linguist'),
96 (1987, 'Economics', 'Robert Solow', 'USA', 'Economist'),
97 (1994, 'Literature', 'Kenzaburo Oe', 'Japan', 'Linguist');
98 • SELECT * FROM Nobel;
99 • SELECT COUNT(*) FROM Nobel;

100

c. From the above table, write a SQL query to find the Nobel Prize winner(s) in the following years (Return year, subject and winner) :

- i. 1970
- ii. 1987

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance sql course Local instance MySQL80

SCHEMAS: analysts employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

Analysts employee_table ibm employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL HW6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

Limit to 10000 rows

Result Grid: Filter Rows: Export: Wrap Cell Content: 15

YEAR SUBJECT WINNER

| YEAR | SUBJECT | WINNER |
|------|------------|------------------------|
| 1970 | Physics | Hannes Alfvén |
| 1970 | Physics | Louis Néel |
| 1970 | Chemistry | Luis Federico Leloir |
| 1970 | Physiology | Bernard Katz |
| 1970 | Literature | Aleksandr Solzhenitsyn |
| 1970 | Economics | Paul Samuelson |
| 1970 | Physiology | Julius Axelrod |

NOBEL 54 x

Output: Action Output

Object Info Session # Time Action Message Duration / Fetch

120 18:37:57 SELECT YEAR, SUBJECT, WINNER FROM NOBEL WHERE YEAR = '1970' LIMIT 0, 10000 6 row(s) returned 0.000 sec / 0.000 sec

Query Completed

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

analysts employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL_Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - sales1
 - sales2
 - Views
 - Stored Procedures
 - Functions
- ablejobs
- cust_omer
- miniproject_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- moviesdb
- mydb
- sakila

Administration Schemas

No object selected

```

110
111      #ii- 1987
112 •  SELECT YEAR, SUBJECT, WINNER
113   FROM NOBEL
114 WHERE YEAR = '1987';
115
116
117
118
119
120
121
122
123
  
```

Result Grid

| YEAR | SUBJECT | WINNER |
|------|------------|------------------------|
| 1987 | Chemistry | Donald J. Cram |
| 1987 | Chemistry | Jean-Marie Lehn |
| 1987 | Physiology | Susumu Tonegawa |
| 1987 | Physics | Johannes Georg Bednorz |
| 1987 | Literature | Joseph Brodsky |
| 1987 | Economics | Robert Solow |

NOBEL 55 x

Output

Action Output

Object Info Session

Time Action Message Duration / Fetch
121 18:39:48 SELECT YEAR, SUBJECT, WINNER FROM NOBEL WHERE YEAR = '1987' LIMIT 0, 10000 8 row(s) returned 0.000 sec / 0.000 sec

Query Completed

d. From the above table, write a SQL query to find the Nobel Prize winner in 'Literature' in the year 1971. Return winner.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

analysts employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL_Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 SQL File 34*

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - sales1
 - sales2
 - Views
 - Stored Procedures
 - Functions
- ablejobs
- cust_omer
- miniproject_db
 - Tables
 - Views
 - Stored Procedures
 - Functions
- moviesdb
- mydb
- sakila

Administration Schemas

No object selected

```

113 WHERE YEAR = '1987';
114
115 /* 4.2.d- From the above table, write a SQL query to find the Nobel Prize winner in 'Literature' in the year 1971.
116 Return winner.*/
117
118 •  SELECT WINNER
119   FROM Nobel
120 WHERE SUBJECT = 'Literature'
121 AND YEAR = '1971';
122
123
124
125
126
  
```

Result Grid

| WINNER |
|--------------|
| Pablo Neruda |

Nobel 58 x

Output

Action Output

Object Info Session

Time Action Message Duration / Fetch
124 18:43:20 SELECT WINNER FROM Nobel WHERE year = '1971' AND subject = 'Literature' LIMIT 0, 10000 1 row(s) returned 0.015 sec / 0.000 sec

Query Completed

e. From the following table, write a SQL query to find the Nobel Prize winner 'Dennis Gabor'. Return year, subject.

The screenshot shows the MySQL Workbench interface. In the top navigation bar, tabs include 'Local instance sql course' and 'Local instance MySQL80'. The main area displays a SQL editor with the following code:

```

123 /* 4.2.e - From the following table, write a SQL query to find the Nobel Prize winner 'Dennis Gabor'. Return year, subject.*/
124
125 • SELECT YEAR, SUBJECT
126 FROM Nobel
127 WHERE WINNER = 'Dennis Gabor';
128
129
130
131
132
133
134
135
136

```

Below the editor is a 'Result Grid' showing the output:

| YEAR | SUBJECT |
|------|---------|
| 1971 | Physics |

At the bottom, the 'Nobel 59' output pane shows the executed query and its results:

```

Object Info Session
# Time Action
125 18:44:35 SELECT WINNER FROM Nobel WHERE SUBJECT = 'Literature' AND YEAR = '1971' LIMIT 0, 10000
Message 1 row(s) returned
Duration / Fetch 0.000 sec / 0.000 sec

```

Query Completed

- f. From the following table, write a SQL query to find the Nobel Prize winners in 'Physics' since the year 1950. Return winner.

The screenshot shows the MySQL Workbench interface. In the top navigation bar, tabs include 'Local instance sql course' and 'Local instance MySQL80'. The main area displays a SQL editor with the following code:

```

129 /* 4.2.f - From the following table, write a SQL query to find the Nobel Prize winners in 'Physics' since the year 1950.
130 Return winner.*/
131
132 • SELECT WINNER
133 FROM Nobel
134 WHERE SUBJECT = 'Physics'
135 AND YEAR >= 1950;
136
137
138
139
140
141
142

```

Below the editor is a 'Result Grid' showing the output:

| WINNER |
|------------------------|
| Hannes Alfvén |
| Louis Neel |
| Dennis Gabor |
| Johannes Georg Bednorz |

At the bottom, the 'Nobel 60' output pane shows the executed query and its results:

```

Object Info Session
# Time Action
126 18:46:55 SELECT YEAR, SUBJECT FROM Nobel WHERE WINNER = 'Dennis Gabor' LIMIT 0, 10000
Message 1 row(s) returned
Duration / Fetch 0.016 sec / 0.000 sec

```

Query Completed

- g. From the following table, write a SQL query to find the Nobel Prize winners in 'Chemistry' between the years 1965 to 1975. Begin and end values are included. Return year, subject, winner, and country

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas

No object selected

```

136
137 /* 4.2.g - From the following table, write a SQL query to find the Nobel Prize winners in 'Chemistry' between the
138 years 1965 to 1975. Begin and end values are included. Return year, subject, winner, and country. */
139
140 • SELECT YEAR, SUBJECT, WINNER, COUNTRY
141 FROM Nobel
142 WHERE SUBJECT = 'Chemistry'
143 AND (YEAR >= 1965 AND YEAR <= 1975);
144
145
146
147
148
149
150
151

```

Result Grid

| YEAR | SUBJECT | WINNER | COUNTRY |
|------|-----------|----------------------|---------|
| 1970 | Chemistry | Luis Federico Leloir | France |
| 1971 | Chemistry | Gerhard Herzberg | Germany |

Nobel 65 x

Action Output

| # | Time | Action | Message | Duration / Fetch |
|-----|----------|---|---|-----------------------|
| 138 | 18:58:30 | SELECT YEAR, SUBJECT, WINNER, COUNTRY FROM Nobel WHERE SUBJECT = 'Chemistry' AND Y... | Error Code: 1146. Table 'able_jobs.nobel' doesn't exist | 0.000 sec / 0.000 sec |
| 139 | 18:58:57 | SELECT YEAR, SUBJECT, WINNER, COUNTRY FROM Nobel WHERE SUBJECT = 'Chemistry' AND Y... | 2 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Query Completed

h. Write a SQL query to show all details of the Prime Ministerial winners after 1972 of Menachem Begin and Yitzhak Rabin.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas

No object selected

```

143 AND (YEAR >= 1965 AND YEAR <= 1975);
144
145 /* 4.2.h - Write a SQL query to show all details of the Prime Ministerial winners after 1972 of Menachem Begin and
146 Yitzhak Rabin.*/
147
148 • SELECT *
149 FROM Nobel
150 WHERE CATEGORY = 'Prime Minister'
151 AND YEAR >= 1972
152 AND WINNER IN ( 'Menachem Begin', 'Yitzhak Rabin' );
153
154
155
156
157
158

```

Result Grid

| YEAR | SUBJECT | WINNER | COUNTRY | CATEGORY |
|------|---------|----------------|---------|----------------|
| 1978 | Peace | Menachem Begin | Israel | Prime Minister |
| 1994 | Peace | Yitzhak Rabin | Israel | Prime Minister |

Nobel 75 x

Action Output

| # | Time | Action | Message | Duration / Fetch |
|-----|----------|--|-------------------|-----------------------|
| 152 | 19:09:13 | SELECT * FROM Nobel WHERE CATEGORY = 'Prime Minister' AND YEAR >= 1972 AND WINNER LIK... | 0 row(s) returned | 0.000 sec / 0.000 sec |
| 153 | 19:09:24 | SELECT * FROM Nobel WHERE CATEGORY = 'Prime Minister' AND YEAR >= 1972 AND WINNER IN (...) | 2 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Query Completed

i. From the following table, write a SQL query to find the details of the winners whose first name matches with the string 'Louis'. Return year, subject, winner, country, and category.

MySQL Workbench

Local instance sql course x Local Instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 nobel

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - nobel
- ablejobs
- cust_omer
- miniproject_db
- Tables

No object selected

```

153
154 /* 4.2.i - From the following table, write a SQL query to find the details of the winners whose first name matches
155   with the string 'Louis'. Return year, subject, winner, country, and category.*/
156
157 • SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY
158   FROM Nobel
159   WHERE WINNER LIKE 'Louis%';
160
161
162
163
164
165
166
167
168

```

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

| YEAR | SUBJECT | WINNER | COUNTRY | CATEGORY |
|------|---------|------------|---------|-----------|
| 1970 | Physics | Louis Neel | France | Scientist |

Nobel 78 x

Output

Action Output

| # | Time | Action | Message | Duration / Fetch |
|-----|----------|--|-------------------|-----------------------|
| 155 | 19:12:37 | SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY FROM Nobel WHERE WINNER LIKE '...' | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 156 | 19:13:04 | SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY FROM Nobel WHERE WINNER LIKE '...' | 1 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Query Completed

- j. From the following table, write a SQL query to find the details of the Nobel Prize winner 'Johannes Georg Bednorz'. Return year, subject, winner, country, and category.

MySQL Workbench

Local instance sql course x Local Instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 nobel

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - nobel
- ablejobs
- cust_omer
- miniproject_db
- Tables

No object selected

```

160
161
162 /* 4.2.j - From the following table, write a SQL query to find the details of the Nobel Prize winner
163   'Johannes Georg Bednorz'. Return year, subject, winner, country, and category.*/
164
165 • SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY
166   FROM Nobel
167   WHERE WINNER = 'Johannes Georg Bednorz';
168
169
170
171
172
173
174
175

```

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

| YEAR | SUBJECT | WINNER | COUNTRY | CATEGORY |
|------|---------|------------------------|---------|-----------|
| 1987 | Physics | Johannes Georg Bednorz | Germany | Scientist |

Nobel 79 x

Output

Action Output

| # | Time | Action | Message | Duration / Fetch |
|-----|----------|---|-------------------|-----------------------|
| 156 | 19:13:04 | SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY FROM Nobel WHERE WINNER LIKE '...' | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 157 | 19:15:39 | SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY FROM Nobel WHERE WINNER = 'Joha...' | 1 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Query Completed

3.

- Create a database with the name: AbleJobs
- Create the following Table with the name: Orders

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', there is a database named 'able_jobs'. Inside this database, a table named 'Orders' is being created. The SQL code for creating the table is as follows:

```
170 • # 4.3
171 • # 4.3.a - Create a database with the name: AbleJobs
172 • USE Able_jobs;
173 • # 4.3.b - Create the Table with the name: Orders
174
175 • CREATE TABLE Orders (
176     ord_no INT PRIMARY KEY,
177     purch_amt FLOAT,
178     ord_date DATE,
179     customer_id INT,
180     salesman_id INT
181 );
182 • INSERT INTO Orders (ord_no, purch_amt, ord_date, customer_id, salesman_id)
183     VALUES
184     (70001, 150.5, '2012-10-05', 3005, 5002),
185     (70009, 270.65, '2012-09-10', 3001, 5005),
186     (70002, 65.26, '2012-10-05', 3002, 5001),
187     (70004, 110.5, '2012-08-17', 3009, 5003),
188     (70007, 948.5, '2012-09-10', 3005, 5002),
189     (70005, 2400.6, '2012-07-27', 3007, 5001),
190     (70008, 5760, '2012-09-10', 3002, 5001),
191     (70010, 1983.43, '2012-10-10', 3004, 5006),
192     (70003, 2488.4, '2012-10-10', 3009, 5003),
193     (70012, 250.45, '2012-06-27', 3008, 5002),
194     (70011, 75.29, '2012-08-17', 3003, 5007),
195     (70013, 3045.6, '2012-04-25', 3002, 5001);
```

- From the following table, write a SQL query to calculate total purchase amount of all orders. Return total purchase amount.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', there is a database named 'able_jobs'. Inside this database, a query is being run to calculate the total purchase amount of all orders. The SQL code is as follows:

```
195 (70013, 3045.6, '2012-04-25', 3002, 5001);
196
197 • SELECT * FROM Orders;
198
199
200 • /*4.3.c - From the following table, write a SQL query to calculate total purchase amount of all orders.
201     Return total purchase amount.*/
202
203 • SELECT SUM(purch_amt) AS "total purchase amount"
204     FROM Orders;
205
206
207
208
209
```

The result grid shows the output of the query:

| total purchase amount |
|-----------------------|
| 17541.180145263672 |

- From the following table, write a SQL query to calculate average purchase amount of all orders. Return average purchase amount.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 nobel

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - nobel
 - Columns
 - Indexes
 - Foreign Key
 - Triggers
 - sales1
 - sales2
 - Views
 - Stored Procedures
 - ablejobs
 - cust_omer
 - miniproject_db
 - Tables

No object selected

```

204     FROM Orders;
205
206  /* 4.3.d - From the following table, write a SQL query to calculate average purchase amount of all orders.
207  Return average purchase amount.*/
208
209 •   SELECT AVG(purch_amt) AS "average purchase amount"
210     FROM Orders;
211
212
213
214
215
216
217
218

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Read Only

average purchase amount
1461.765012105306

Object Info Session Output

Query Completed

e. From the following table, write a SQL query to count the number of unique salespeople.
Return number of salespeople.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 nobel

SCHEMAS

- able_homeworks
- able_jobs
 - Tables
 - nobel
 - Columns
 - Indexes
 - Foreign Key
 - Triggers
 - sales1
 - sales2
 - Views
 - Stored Procedures
 - ablejobs
 - cust_omer
 - miniproject_db
 - Tables

No object selected

```

210     FROM Orders;
211
212  /* 4.3.e - From the following table, write a SQL query to count the number of unique salespeople.
213  Return number of salespeople.*/
214
215 •   SELECT COUNT(DISTINCT(salesman_id))
216     AS "number of salespeople"
217     FROM Orders;
218
219
220
221
222
223
224

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Read Only

number of salespeople
6

Object Info Session Output

Query Completed

f. From the following table, write a SQL query to count the number of customers.
Return number of customers.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 nobel
Schemas
employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 nobel
219 /* 4.3.f - From the following table, write a SQL query to count the number of customers. Return number of customers.*/
220
221 • SELECT COUNT(*)
222 AS "number of customers"
223 FROM Orders;
224
225
226
227
228
229
230
231
232
233
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
number of customers
12
Result 91 x
Object Info Session Output
Query Completed

```

Session 5: Union and Join

- 1.
- a. Create a database with the name: AbleJobs
- b. Create the following Table with the name: Nobel
- c. From the above table, write a SQL query to combine the winners in Physics, 1970 and in Economics, 1971. Return year, subject, winner, country, and category.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 Session5* nobel
Schemas
employee_table ibm_employees - sheet1 ibm_emp SQL File 22* SQL File 27* SQL Hw6* SQL File 29 SQL File 30* SQL_miniproject Session2 Session3 Session4 Session5* nobel
1 # Session5: Union and Join
2 # 5.1.a - Create a database with the name: AbleJobs
3 • USE Able_jobs;
4 #5.1.b - Create the Table with the name: Nobel
5 • SELECT * FROM Nobel;
6
7 /* 5.1.c - From the above table, write a SQL query to combine the winners in Physics, 1970 and in Economics,
8 1971. Return year, subject, winner, country, and category.*/
9
10 • SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY
11 FROM Nobel
12 WHERE (SUBJECT = 'Physics' AND YEAR = 1970)
13 UNION
14 (SELECT YEAR, SUBJECT, WINNER, COUNTRY, CATEGORY
15 FROM Nobel
16 WHERE (SUBJECT = 'Economics' AND YEAR = 1971))
17
18
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
YEAR SUBJECT WINNER COUNTRY CATEGORY
1970 Physics Hannes Alfvén Sweden Scientist
1970 Physics Louis Neel France Scientist
1971 Economics Simon Kuznets Russia Economist
Result 2 x
Object Info Session Output
Action Output
# Time Action
168 19:43:21 SELECT * FROM Orders LIMIT 0, 10000
Message 12 row(s) returned
Duration / Fetch 0.000 sec / 0.000 sec
Query Completed

```

2.

- a. Create a database with the name: AbleJobs
- b. Create the following Table with the name: Sales2
- c. Create the following table with the name: Sales1
- d. From the above tables write a SQL query to find the salesperson and customer who belongs to same city. Return Salesman, cust_name and city.

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance sql course' and 'Local instance MySQL80' are selected. The 'Schemas' tab is active, showing the 'able_jobs' schema with its tables: 'nobel', 'sales1', and 'sales2'. The 'Tables' section is expanded. The central pane displays a SQL script:

```
20  # 5.2
21  #5.2.a - Create a database with the name: AbleJobs
22 • USE Able_jobs;
23
24 • SELECT * FROM Sales2;# customer
25
26 • SELECT * FROM Sales1;# salesman
27
28 /* 5.2.d - From the above tables write a SQL query to find the salesperson and customer who belongs to same city.
29  Return Salesman, cust_name and city.*/
30
31 • SELECT Sales2.cust_name AS Customer,
32   Sales2.name AS Salesman,
33   Sales1.city
34   FROM Sales1
35   INNER JOIN Sales2
36   ON Sales1.city = Sales2.city;
```

The 'Result Grid' pane below shows the query results:

| Customer | Salesman | city |
|----------------|------------|----------|
| Nick Rimando | James Hoog | New York |
| Fabian Johnson | Mc Lyon | Paris |
| Fabian Johnson | Lauson Hen | Paris |
| Fabian Johnson | Nial Krite | Paris |
| Brad Davis | James Hoog | New York |
| Julian Green | Pit Alex | London |
| Geoff Cameron | Pit Alex | London |

- e. From the above tables write a SQL query to find the salesperson(s) and the customer(s) he handle. Return Customer Name, city, Salesman, commission.

The screenshot shows the MySQL Workbench interface with the same session setup. The 'Schemas' tab is active, showing the 'able_jobs' schema with its tables: 'nobel', 'sales1', and 'sales2'. The 'Tables' section is expanded. The central pane displays a SQL script:

```
34   FROM Sales1
35   INNER JOIN Sales2
36   ON Sales1.city = Sales2.city;
37
38 /* 5.2.e - From the above tables write a SQL query to find the salesperson(s) and the customer(s) he handle.
39  Return Customer Name, city, Salesman, commission.*/
40
41 • SELECT Sales2.cust_name AS "Customer Name",
42   Sales2.city, Sales1.name AS "Salesman", Sales1.commission
43   FROM Sales2
44   INNER JOIN Sales1
45   ON Sales1.salesman_id = Sales2.salesman_id;
46
47
48
49
50
```

The 'Result Grid' pane below shows the query results:

| Customer Name | city | Salesman | commission |
|----------------|------------|------------|------------|
| Nick Rimando | New York | James Hoog | 0.15 |
| Zoey Aldor | Moscow | Paul Adam | 0.13 |
| Fabian Johnson | Paris | Mc Lyon | 0.14 |
| Graham Zusi | California | Nial Krite | 0.13 |
| Brad Davis | New York | James Hoog | 0.15 |
| Julian Green | London | Nial Krite | 0.13 |
| Geoff Cameron | London | Lauson Hen | 0.12 |

f. From the above tables write a SQL query to find those salespersons who received a commission from the company more than 12%.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

45  ON Sales1.salesman_id = Sales2.salesman_id;
46
47  /* 5.2.f - From the above tables write a SQL query to find those salespersons who received a commission from
48  the company more than 12%.*/
49
50 •  SELECT* FROM Sales1;
51 •  SELECT name AS Salespersons, commission
52  FROM Sales1
53  WHERE commission > 0.12;
54
55
56
57
58
59
60

```

The Result Grid shows the following data:

| Salespersons | commission |
|--------------|------------|
| James Hoog | 0.15 |
| Nail Knite | 0.13 |
| Pit Alex | 0.22 |
| Mc Lyon | 0.14 |
| Paul Adam | 0.13 |

g. From the following tables write a SQL query to find those salespersons do not live in the same city where their customers live and received a commission from the company more than 12%. Return Customer Name, customer city, Salesman, salesman city, commission.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

54
55  /* 5.2.g - From the following tables write a SQL query to find those salespersons do not live in the same city
56  where their customers live and received a commission from the company more than 12%. Return Customer Name, customer city,
57  Salesman, salesman city, commission.*/
58
59 •  SELECT Sales2.cust_name AS "Customer Name",
60  Sales2.city, Sales1.name AS Salesman,
61  Sales1.city, Sales1.commission
62  FROM Sales2
63  INNER JOIN Sales1
64  ON Sales2.salesman_id = Sales1.salesman_id
65  WHERE Sales1.commission > 0.12
66  AND Sales1.city != Sales2.city ;
67
68
69
70

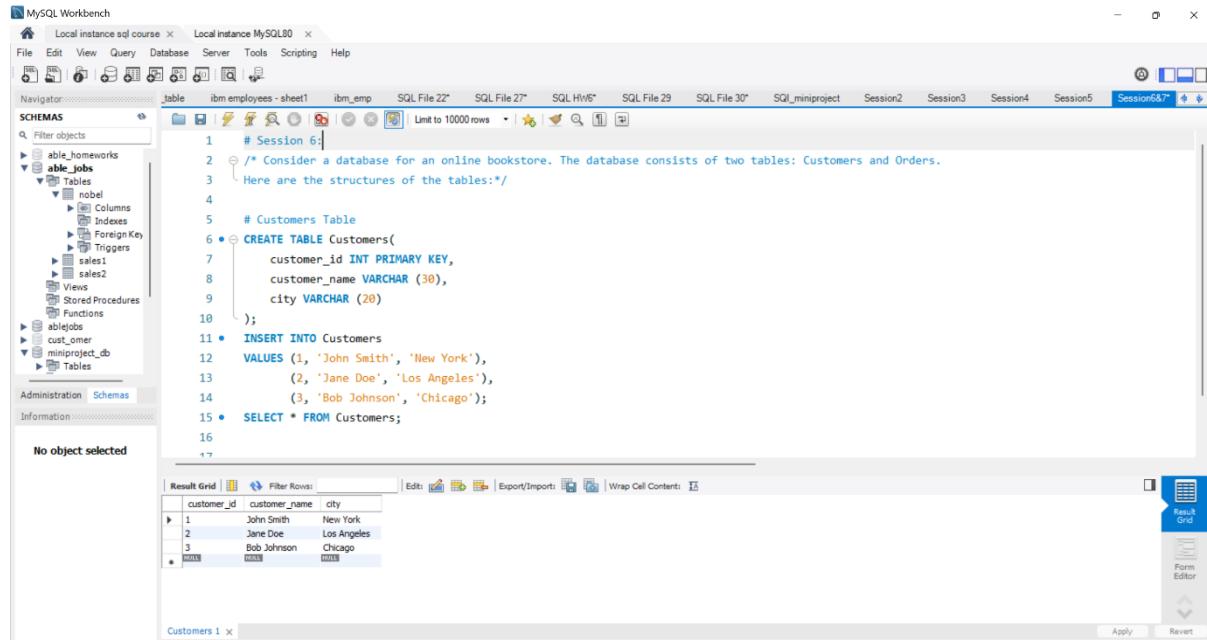
```

The Result Grid shows the following data:

| Customer Name | city | Salesman | city | commission |
|---------------|------------|------------|-------|------------|
| Zoey Aldor | Moscow | Paul Adam | Rome | 0.13 |
| Graham Zusi | California | Nail Knite | Paris | 0.13 |
| Julian Green | London | Nail Knite | Paris | 0.13 |

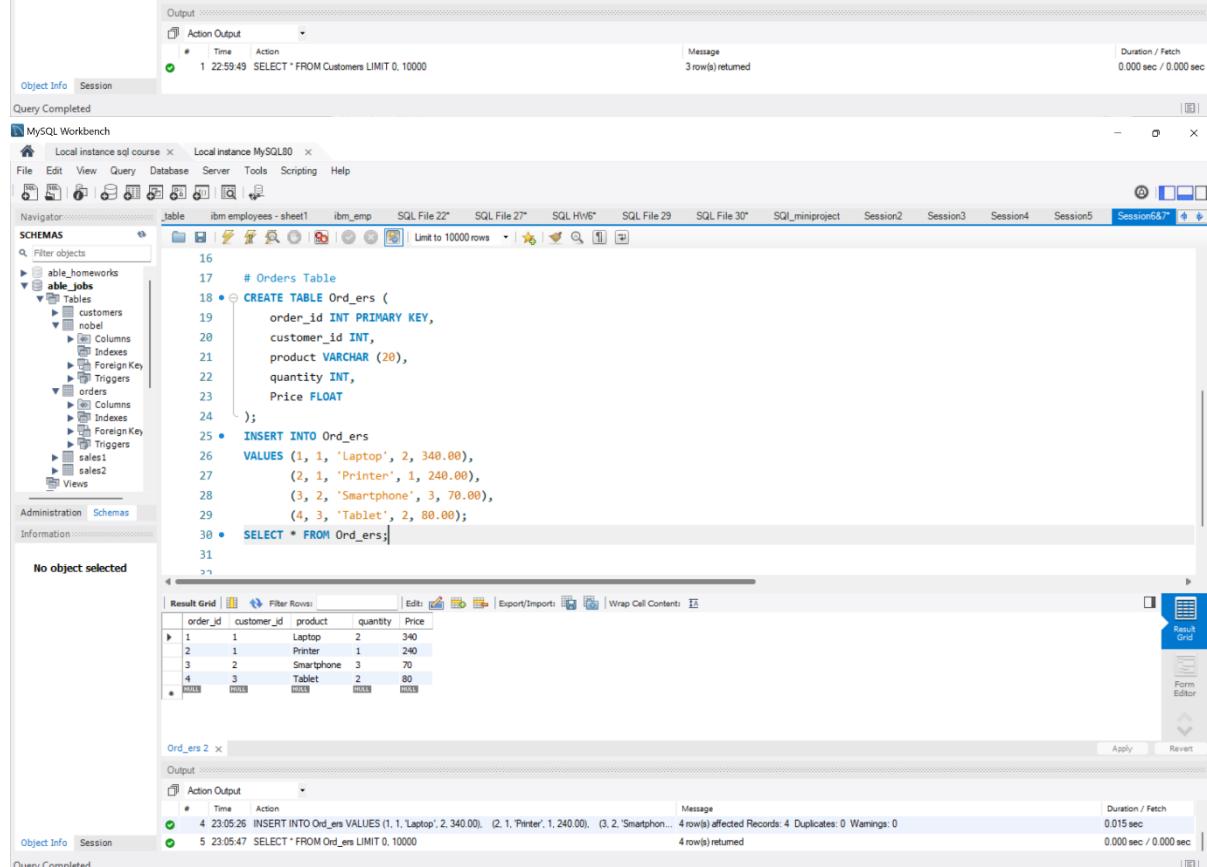
Session 6 & 7: Nested Queries & Normalization

Consider a database for an online bookstore. The database consists of two tables: Customers and Orders. Here are the structures of the tables:



The screenshot shows the MySQL Workbench interface with the 'Customers' table structure defined in the script pane. The table has columns: customer_id (INT PRIMARY KEY), customer_name (VARCHAR(30)), and city (VARCHAR(20)). Data is inserted into the table with three rows: (1, 'John Smith', 'New York'), (2, 'Jane Doe', 'Los Angeles'), and (3, 'Bob Johnson', 'Chicago'). A query is run to select all data from the Customers table.

| customer_id | customer_name | city |
|-------------|---------------|-------------|
| 1 | John Smith | New York |
| 2 | Jane Doe | Los Angeles |
| 3 | Bob Johnson | Chicago |
| * | NULL | NULL |



The screenshot shows the MySQL Workbench interface with the 'Orders' table structure defined in the script pane. The table has columns: order_id (INT PRIMARY KEY), customer_id (INT), product (VARCHAR(20)), quantity (INT), and Price (FLOAT). Data is inserted into the table with four rows: (1, 1, 'Laptop', 2, 340.00), (2, 1, 'Printer', 1, 240.00), (3, 2, 'Smartphone', 3, 70.00), and (4, 3, 'Tablet', 2, 80.00). A query is run to select all data from the Orders table.

| order_id | customer_id | product | quantity | Price |
|----------|-------------|------------|----------|-------|
| 1 | 1 | Laptop | 2 | 340 |
| 2 | 1 | Printer | 1 | 240 |
| 3 | 2 | Smartphone | 3 | 70 |
| 4 | 3 | Tablet | 2 | 80 |
| * | NULL | NULL | NULL | NULL |

Question 1:

Retrieve the names of all customers who have placed an order for a product with a price greater than \$100.

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

- Navigator:** Shows the schema structure for the 'able_jobs' database, including tables like 'customers', 'nobel', 'orders', and 'sales1'.
- SQL Editor:** Contains the following SQL code:

```
29      (4, 3, 'Tablet', 2, 80.00);
30 •  SELECT * FROM Ord_ers;
31
32 /* Question 1:
33   Retrieve the names of all customers who have placed an order for a product with a price greater than $100.*/
34
35 •  SELECT customer_name
36   FROM Customers
37   WHERE customer_id
38   IN (SELECT customer_id
39   FROM Ord_ers
40   WHERE Price > 100
41 );
42
43
44
```
- Result Grid:** Displays the result of the query, showing a single row: "customer_name" with value "John Smith".
- Output:** Shows the execution log with two entries:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|---|---------|-----------------------|
| 19 | 23:25:29 | SELECT customer_name FROM Customers WHERE customer_id IN (SELECT customer_id FROM Ord_ers... 1 row(s) returned) | | 0.000 sec / 0.000 sec |
| 20 | 23:33:52 | SELECT customer_name FROM Customers WHERE customer_id IN (SELECT customer_id FROM Ord_ers... 1 row(s) returned) | | 0.000 sec / 0.000 sec |

Customers 8 x

Object Info **Session**

Query Completed

Question 2:

List the products that have been ordered by customers from the same city as customer 'John Smith'.

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

- Navigator:** Shows the schema structure for the 'able_jobs' database, including tables like 'customers', 'nobel', 'orders', and 'sales1'.
- SQL Editor:** Contains the following SQL code:

```
40   WHERE Price > 100
41 );
42
43 /* Question 2:
44   List the products that have been ordered by customers from the same city as customer 'John Smith'. */
45
46 •  SELECT DISTINCT product
47   FROM Ord_ers
48   WHERE customer_id IN (
49   SELECT customer_id
50   FROM Customers
51   WHERE city = (SELECT city FROM Customers WHERE customer_name = 'John Smith')
52 );
53
54
55
```
- Result Grid:** Displays the result of the query, showing two rows: "product" with values "Laptop" and "Printer".
- Output:** Shows the execution log with two entries:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|---------|-----------------------|
| 24 | 23:40:14 | SELECT DISTINCT product FROM Ord_ers WHERE customer_id IN (SELECT customer_id FROM Customers WHERE customer_name = 'John Smith') 2 row(s) returned | | 0.000 sec / 0.000 sec |
| 25 | 23:40:22 | SELECT product FROM Ord_ers WHERE customer_id IN (SELECT customer_id FROM Customers WHERE customer_name = 'John Smith') 2 row(s) returned | | 0.000 sec / 0.000 sec |

Ord_ers 12 x

Object Info **Session**

Query Completed

Question 3:

Find the order IDs and total order amounts for orders that contain at least one product with a quantity greater than 2.

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

```
49  SELECT customer_id
50  FROM Customers
51  WHERE city = (SELECT city FROM Customers WHERE customer_name = 'John Smith')
52  ;
53
54  /* Question 3:
55  Find the order IDs and total order amounts for orders that contain at least one product with a quantity greater than 2. */
56
57 •  SELECT order_id, SUM(Price * quantity) AS 'total_order_amount'
58  FROM Orders
59  GROUP BY order_id
60  HAVING MAX(quantity) > 2;
61
62
63
64
rr
```

The results grid shows the following data:

| order_id | total_order_amount |
|----------|--------------------|
| 3 | 210 |

The output pane displays the execution log:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|-------------------|-----------------------|
| 32 | 23:59:18 | SELECT order_id, SUM(Price * quantity) AS 'total_order_amount' FROM Orders GROUP BY order_id HAVING MAX(quantity) > 2; | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 33 | 00:00:35 | SELECT order_id, SUM(Price * quantity) AS 'total_order_amount' FROM Orders GROUP BY order_id HAVING MAX(quantity) > 2; | 1 row(s) returned | 0.000 sec / 0.000 sec |