

Q1. List customers with no bookings.

```
SELECT DISTINCT CONCAT(CustFirstName,"",CustLastName) AS  
Customer_name,CustStreetAddress,CustCity,CustState,  
CustZipCode,CustAreaCode,CustPhoneNumber  
FROM customers cust  
LEFT JOIN orders ord ON cust.CustomerID=ord.CustomerID  
WHERE ord.CustomerID is NULL  
ORDER BY cust.CustomerID;
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'schemas' tree with 'salesordersexample' expanded, showing tables like 'customers' and 'orders'. The main editor window contains a SQL script with three queries. The first query, '1. List customers with no bookings.', is highlighted in blue. Below the script, the 'Result Grid' shows the output of the first query, displaying customer information for 'JeffreyTrekler'. The bottom status bar shows the 'Output' tab with two messages: 'SELECT vend.VendorID,vend.VendName,vend.ProductName, COUNT(vend.VendName) AS VendCount FR...' (10 row(s) returned) and 'SELECT DISTINCT CONCAT(CustFirstName,"",CustLastName) AS Customer_name,CustStreetAddress,CustCr...' (1 row(s) returned).

```
-- 1. List customers with no bookings.  
SELECT DISTINCT CONCAT(CustFirstName,"",CustLastName) AS Customer_name,CustStreetAddress,CustCity,CustState,  
CustZipCode,CustAreaCode,CustPhoneNumber  
FROM customers cust  
LEFT JOIN orders ord ON cust.CustomerID=ord.CustomerID  
WHERE ord.CustomerID is NULL  
ORDER BY cust.CustomerID;
```

Customer_name	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhoneNumber
JeffreyTrekler	19541 104th Ave NE	Bothell	WA	98006	425	555-9999

Result 16 x

Table: customers

Columns:

- CustomerID int(11) PK
- CustFirstName varchar(25)
- CustLastName varchar(25)

Object Info Session

Output

#	Time	Action	Message
108	13:51:41	SELECT vend.VendorID,vend.VendName,vend.ProductName, COUNT(vend.VendName) AS VendCount FR...	10 row(s) returned
109	13:55:30	SELECT DISTINCT CONCAT(CustFirstName,"",CustLastName) AS Customer_name,CustStreetAddress,CustCr...	1 row(s) returned

Q2. Display all tournaments and any matches that have been played.

```
SELECT TourneyDate,TourneyLocation, MatchID
```

```
FROM tournaments tour
```

```
INNER JOIN tourney_matches tourmat ON tour.TourneyID= tourmat.TourneyID;
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'bowlingleagueexample' database schema with tables like bowler_scores, bowlers, match_games, teams, tournaments, and tourney_matches. The central SQL editor contains three queries. The second query, which is highlighted, is:

```
SELECT TourneyDate,TourneyLocation, MatchID
FROM tournaments tour
INNER JOIN tourney_matches tourmat ON tour.TourneyID= tourmat.TourneyID;
```

 The 'Result Grid' at the bottom shows 10 rows of data. The bottom status bar indicates that the query was executed successfully, returning 57 rows.

TourneyDate	TourneyLocation	MatchID
2012-09-04	Red Rooster Lanes	1
2012-09-04	Red Rooster Lanes	2
2012-09-04	Red Rooster Lanes	3
2012-09-04	Red Rooster Lanes	4
2012-09-11	Thunderbird Lanes	5
2012-09-11	Thunderbird Lanes	6
2012-09-11	Thunderbird Lanes	7
2012-09-11	Thunderbird Lanes	8
2012-09-18	Bolero Lanes	9
2012-09-18	Bolero Lanes	10

Q3. Produce a list of customers who like contemporary music together with a list of entertainers who play contemporary music.

```
SELECT 'Customer' Cust_Ent,  
CONCAT(customers.CustFirstName,"",customers.CustLastName) as Name, musical_styles.StyleName  
FROM customers  
INNER JOIN musical_preferences ON customers.CustomerID =  
musical_preferences.CustomerID  
INNER JOIN musical_styles ON musical_preferences.StyleID = musical_styles.StyleID  
WHERE musical_styles.StyleName = 'Contemporary'  
  
UNION  
  
SELECT 'Entertainer' Cus_Ent, entertainers.EntStageName as Name, musical_styles.StyleName  
FROM entertainers  
INNER JOIN entertainer_styles ON entertainers.EntertainerID = entertainer_styles.EntertainerID  
INNER JOIN musical_styles ON entertainer_styles.StyleID = musical_styles.StyleID  
WHERE musical_styles.StyleName = 'Contemporary';
```

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

bowlingleagueexample
edw_interface_table
entertainmentagencyexample

Tables

agents
customers
Columns
Indexes
Foreign Keys
Triggers
engagements
entertainer_members
entertainer_styles
Columns
Indexes
Foreign Keys
Triggers
entertainers
members
musical_preferences
Columns
Indexes
Foreign Keys
Triggers
musical_styles
ztbldays
ztblmonths
ztblskiplabels
ztblweeks
Views
Stored Procedures

Administration Schemas

Information

Schema: entertainmentagencyexample

Object Info Session

SQL File 4* SQL File 5* Week6 HW3 SQL for Multiple tables bowler_scores bowlers teams match_games 26th March-SQL* HW4_Complex

Limit to 5000 rows

```
13 FROM tournaments tour  
14 INNER JOIN touney_matches tourmat ON tour.TourneyID= tourmat.TourneyID;  
15  
16 -- 3. Produce a list of customers who like contemporary music together with a list of entertainers who play contemporary music.  
17  
18 SELECT 'Customer' Cust_Ent, CONCAT(customers.CustFirstName,"",customers.CustLastName) as Name, musical_styles.StyleName  
19 FROM customers  
20 INNER JOIN musical_preferences ON customers.CustomerID = musical_preferences.CustomerID  
21 INNER JOIN musical_styles ON musical_preferences.StyleID = musical_styles.StyleID  
22 WHERE musical_styles.StyleName = 'Contemporary'  
23  
24 UNION  
25 SELECT 'Entertainer' Cus_Ent, entertainers.EntStageName as Name, musical_styles.StyleName  
26 FROM entertainers  
27 INNER JOIN entertainer_styles ON entertainers.EntertainerID = entertainer_styles.EntertainerID  
28 INNER JOIN musical_styles ON entertainer_styles.StyleID = musical_styles.StyleID  
29 WHERE musical_styles.StyleName = 'Contemporary';  
30  
31 -- 4. List customers who have booked entertainers who play country or country rock.
```

Result Grid

	Cust_Ent	Name	StyleName
▶	Customer	DorisHartwig	Contemporary
▶	Customer	DarrenGehring	Contemporary
▶	Customer	KerryPatterson	Contemporary
▶	Entertainer	Carol Peacock Trio	Contemporary
▶	Entertainer	Caroline Cole Cuartet	Contemporary

Result 2 x

Output

Action Output

#	Time	Action	Message
1	13:59:42	SELECT TouneyDate, TouneyLocation, MatchID FROM tournaments tour INNER JOIN touney_matches tour...	57 row(s) returned
2	14:00:51	SELECT 'Customer' Cust_Ent, CONCAT(customers.CustFirstName,"",customers.CustLastName) as Name, musica...	5 row(s) returned

Read Only

Q4. List customers who have booked entertainers who play country or country rock.

```
SELECT CONCAT (CustFirstName,',',CustLastName) AS CustName,EntStageName,StyleName
```

```
FROM customers
```

```
INNER JOIN engagements ON customers.CustomerID=engagements.CustomerID
```

```
INNER JOIN entertainers ON engagements.EntertainerID=entertainers.EntertainerID
```

```
INNER JOIN entertainer_styles ON entertainers.EntertainerID = entertainer_styles.EntertainerID
```

```
INNER JOIN musical_styles ON entertainer_styles.StyleID= musical_styles.StyleID
```

```
WHERE StyleName = 'Country' OR StyleName = 'Country Rock'
```

```
ORDER BY CustName;
```

The screenshot shows the MySQL Workbench interface. The SQL Editor contains a query that joins the customers, engagements, entertainers, entertainer_styles, and musical_styles tables to find customers who have booked entertainers playing country or country rock. The query is as follows:

```
25 FROM entertainers
26
27 INNER JOIN entertainer_styles ON entertainers.EntertainerID = entertainer_styles.EntertainerID
28 INNER JOIN musical_styles ON entertainer_styles.StyleID = musical_styles.StyleID
29 WHERE musical_styles.StyleName = 'Contemporary';
30
31 -- 4.List customers who have booked entertainers who play country or country rock.
32
33 SELECT CONCAT (CustFirstName,',',CustLastName) AS CustName,EntStageName,StyleName
34 FROM customers
35
36 INNER JOIN engagements ON customers.CustomerID=engagements.CustomerID
37 INNER JOIN entertainers ON engagements.EntertainerID=entertainers.EntertainerID
38 INNER JOIN entertainer_styles ON entertainers.EntertainerID = entertainer_styles.EntertainerID
39 INNER JOIN musical_styles ON entertainer_styles.StyleID= musical_styles.StyleID
40 WHERE StyleName = 'Country' OR StyleName = 'Country Rock'
41 ORDER BY CustName;
42
43 -- 5.Display students enrolled in a class on Tuesday.
44
45 SELECT DISTINCT StudFirstName,StudSettings,CONCAT(StudFirstName,' ',StudSettings) AS StudSettings
```

The Results window shows the following data:

CustName	EntStageName	StyleName
CarolViescas	Country Feeling	Country
DeanMcCrae	Country Feeling	Country
DebWaldal	Country Feeling	Country
DebWaldal	Country Feeling	Country
DebWaldal	Coldwater Cattle Company	Country
DebWaldal	Coldwater Cattle Company	Country Rock
DebWaldal	Coldwater Cattle Company	Country
DebWaldal	Coldwater Cattle Company	Country Rock
DorishHartwig	Coldwater Cattle Company	Country
DorishHartwig	Coldwater Cattle Company	Country Rock

The Action Output window shows the following messages:

#	Time	Action	Message
2	14:00:51	SELECT 'Customer' Cust_Ent, CONCAT(customers.CustFirstName,',',customers.CustLastName) as Name, musi...	5 row(s) returned
3	14:01:35	SELECT CONCAT (CustFirstName,',',CustLastName) AS CustName,EntStageName,StyleName FROM custome...	31 row(s) returned

Q5.Display students enrolled in a class on Tuesday.

```
SELECT DISTINCT StudFirstName,StudLastName,CONCAT(StudFirstName,' ',StudLastName) AS
StudentName,
classes.ClassID,classes.TuesdaySchedule
FROM students STUD
INNER JOIN student_schedules STUDSCHED ON STUD.StudentID=STUDSCHED.StudentID
INNER JOIN classes ON STUDSCHED.ClassID=classes.ClassID
WHERE classes.TuesdaySchedule=1
ORDER BY STUD.StudentID;
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'schoolschedulingexample' selected. The main editor window contains a SQL query that has been executed. The query is as follows:

```
SELECT CONCAT (CustFirstName,' ',CustLastName) AS CustName,EntStageName,StyleName
FROM customers
INNER JOIN engagements ON customers.CustomerID=engagements.CustomerID
INNER JOIN entertainers ON engagements.EntertainerID=entertainers.EntertainerID
INNER JOIN entertainer_styles ON entertainers.EntertainerID = entertainer_styles.EntertainerID
INNER JOIN musical_styles ON entertainer_styles.StyleID= musical_styles.StyleID
WHERE StyleName = 'Country' OR StyleName = 'Country Rock'
ORDER BY CustName;
```

Below the query, the 'Result Grid' is displayed, showing the following data:

CustName	EntStageName	StyleName
CarolViescas	Country Feeling	Country
DeanMcCrae	Country Feeling	Country
DebWaldal	Country Feeling	Country
DebWaldal	Country Feeling	Country
DebWaldal	Coldwater Cattle Company	Country
DebWaldal	Coldwater Cattle Company	Country Rock
DebWaldal	Coldwater Cattle Company	Country
DebWaldal	Coldwater Cattle Company	Country Rock
DorishHartwig	Coldwater Cattle Company	Country
DorishHartwig	Coldwater Cattle Company	Country Rock

The bottom of the interface shows the 'Table: subjects' with columns: SubjectID (int(11) PK), CategoryID (varchar(10)), and SubjectCode (varchar(8)). The 'Output' pane shows the execution of the query, with a message indicating that 31 row(s) were returned.

Q7.List all vendors and the count of products sold by each. (use a subquery)

SELECT vend.VendorID,vend.VendName,vend.ProductName, COUNT(vend.VendName) AS VendCount

FROM (SELECT vendors.VendorID, vendors.VendName,products.ProductName

FROM vendors

INNER JOIN product_vendors ON vendors.VendorID = product_vendors.VendorID

INNER JOIN products ON product_vendors.ProductNumber = products.ProductNumber) AS vend

GROUP BY vend.VendName

ORDER BY Vend.VendorID;

The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that lists vendors and the count of products sold by each, using a subquery. The query is as follows:

```
46 INNER JOIN student_schedules STUOSCHED ON STUD.StudentID=STUOSCHED.StudentID
47 INNER JOIN classes ON STUOSCHED.ClassID=classes.ClassID
48 WHERE classes.TuesdaySchedule=1
49 ORDER BY STUD.StudentID;
50
51 -- 6.List the ingredients that are used in some recipe where the measurement amount in the recipe is not the default measurement amount
52 -- 7.List all vendors and the count of products sold by each. (use a subquery)
53 SELECT vend.VendorID,vend.VendName,vend.ProductName, COUNT(vend.VendName) AS VendCount
54 FROM (SELECT vendors.VendorID, vendors.VendName,products.ProductName
55 FROM vendors
56 INNER JOIN product_vendors ON vendors.VendorID = product_vendors.VendorID
57 INNER JOIN products ON product_vendors.ProductNumber = products.ProductNumber) AS vend
58 GROUP BY vend.VendName
59 ORDER BY Vend.VendorID;
60
61 -- 8.List each staff member and the count of classes each is scheduled to teach.
62 -- 9.Show me the subject categories that have fewer than three full professors teaching that subject.
63
64 SELECT CategoryDescription, COUNT(DISTINCT StaffID) as professor count
```

The results are displayed in the Result Grid, showing 10 rows of data:

VendorID	VendName	ProductName	VendCount
1	Shinoman, Incorporated	Shinoman 105 SC Brakes	3
2	Viscount	Viscount Mountain Bike	6
3	Nikoma of America	Nikoma Lok-Tight U-Lock	5
4	Proformance	Trek 9000 Mountain Bike	3
5	Kona, Incorporated	Ultra-2K Competition Tire	1
6	Big Sky Mountain Bikes	Eagle FS-3 Mountain Bike	22
7	Dog Ear	Dog Ear Cyclecomputer	9
8	Sun Sports Suppliers	Kryptonite Advanced 2000 U-Lock	5
9	Lone Star Bike Supply	Trek 9000 Mountain Bike	30
10	Armadillo Brand	Pro-Sport Dillo Shades	6

The bottom panel shows the Action Output, indicating that the query was executed successfully and returned 10 rows.

Q8. List each staff member and the count of classes each is scheduled to teach.

```
SELECT fac.fac_Name,COUNT(DISTINCT fac.ClassID) AS CountClass,fac.ClassID
FROM (SELECT CONCAT(staff.StfFirstName,"",staff.StfLastname) AS fac_Name,classes.ClassID
FROM staff
INNER JOIN faculty_classes ON faculty_classes.StaffID=staff.StaffID
INNER JOIN classes ON faculty_classes.ClassID=classes.ClassID) AS fac
GROUP BY fac.fac_Name;
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree for 'schoolschedulingexample', with tables like 'buildings', 'categories', 'class_rooms', 'classes', 'departments', 'faculty', 'faculty_categories', 'faculty_classes', 'faculty_subjects', 'majors', and 'staff'. The main editor shows a SQL query with two parts: a subquery for vendors and a main query for staff and classes. The query is highlighted in blue. The 'Result Grid' at the bottom shows the output of the query, with columns 'fac_Name', 'CountClass', and 'ClassID'. The results are as follows:

fac_Name	CountClass	ClassID
AlainaHallmark	8	4500
AlastairBlack	6	2520
AnnPatterson	8	4180
CalebViescas	8	5103
CarolViescas	4	5917
DavidSmith	6	3045
DebWaldal	10	2911
GaryHallmark	4	5510
JimGlynn	6	5051
JimWilson	6	2451

The bottom of the screenshot shows the 'Action Output' pane with two messages: 'SELECT CONCAT (CustFirstName,"",CustLastName) AS CustName,EntStageName,StyleName FROM custome... 31 row(s) returned' and 'SELECT fac.fac_Name,COUNT(DISTINCT fac.ClassID) AS CountClass,fac.ClassID FROM (SELECT CONC... 22 row(s) returned'.

Q9.Show me the subject categories that have fewer than three full professors teaching that subject.

SELECT CategoryDescription, COUNT(DISTINCT StaffID)as professor_count

FROM categories CAT

JOIN subjects S ON CAT.CategoryID=S.CategoryID

JOIN classes C ON S.SubjectID=C.SubjectID

JOIN faculty_classes FC ON C.ClassID=FC.ClassID

GROUP BY CategoryDescription

HAVING COUNT(DISTINCT StaffID)<3;

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHOOLSCHEDULINGEXAMPLE' database schema with tables like 'categories', 'subjects', 'classes', and 'faculty_classes'. The main editor window contains a SQL script with two queries. The first query (lines 61-67) lists staff members and the count of classes they teach. The second query (lines 72-78) is the one specified in the text, which filters for subject categories with fewer than three full professors. The 'Result Grid' at the bottom shows the output of the second query, displaying four rows of subject categories and their corresponding professor counts.

CategoryDescription	professor_count
Computer Science	2
Geography	1
Music	2
Political Science	2

The 'Output' pane at the bottom shows the execution log, indicating that the second query returned 4 rows.

Q10. List the last name and first name of every bowler whose average raw score is greater than or equal to the overall average score.

SELECT bowlers.BowlerFirstName, bowlers.BowlerLastName

FROM bowlers

INNER JOIN bowler_scores ON bowlers.BowlerID=bowler_scores.BowlerID

GROUP BY bowlers.BowlerID

HAVING avg(bowler_scores.RawScore)>=

(SELECT avg(bowler_scores.RawScore)

FROM bowler_scores);

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'bowlingleagueexample' schema with tables 'bowler_scores', 'bowlers', 'match_games', and 'teams'. The main editor window contains a SQL query for Q10. The 'Result Grid' at the bottom shows the results of the query, which is a table with two columns: 'CategoryDescription' and 'professor_count'.

CategoryDescription	professor_count
Computer Science	2
Geography	1
Music	2
Political Science	2

The 'Output' pane at the bottom shows the execution of the query, indicating that 22 rows were returned.