

SQL ASSIGNMENT SET 1

Sample Dataset 1

Q1. Query all columns for all American cities in the CITY table with populations larger than 100000. The Country-Code for America is USA.

Sol) **select * from city where CountryCode='USA' and population>100000;**

First of all, I have created a database named ‘assignment’ and then created a table inside it named ‘City’. Then Inserted records using insert functions and then performed the desired query.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', there is a tree view of databases and tables. The 'Assignment' database is selected. Inside 'Assignment', there is a 'city' table. A SQL editor window titled 'SQL File 4' contains the following code:

```
3 •  create table city(
4   ID int(10),
5   Name varchar(17),
6   CountryCode varchar(3),
7   District varchar(20),
8   population bigint);
9
10 •  insert into city values
11   ('Rotterdam','NLD','Zuid-Holland',593321),
12   ('Scottsdale','USA','Arizona',202705),
13   ('Corona','USA','California',124966),
14   ('Concord','USA','California',121780),
15   ('Cedar Rapids','USA','Iowa',120758),
16   ('Coral Springs','USA','Florida',117549),
17   ('Farfield','USA','California',92256),
18   ('Boulder','USA','Colorado',91238),
19   ('Fall River','USA','Massachusetts',90555);
```

The right side of the interface has a 'SQLAdditions' panel with the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The screenshot shows the MySQL Workbench interface. The 'Assignment' schema is selected. A SQL editor window titled 'SQL File 4' contains the following code:

```
13   ('Corona','USA','California',124966),
14   ('Concord','USA','California',121780),
15   ('Cedar Rapids','USA','Iowa',120758),
16   ('Coral Springs','USA','Florida',117549),
17   ('Farfield','USA','California',92256),
18   ('Boulder','USA','Colorado',91238),
19   ('Fall River','USA','Massachusetts',90555);
20
21 •  select * from city;
22 •  select * from city where CountryCode='USA' and population>100000;
```

The results are displayed in a 'Result Grid' tab. The grid shows the following data:

ID	Name	CountryCode	District	population
3878	Scottsdale	USA	Arizona	202705
3965	Corona	USA	California	124966
3973	Concord	USA	California	121780
3977	Cedar Rapids	USA	Iowa	120758
3982	Coral Springs	USA	Florida	117549

The right side of the interface has a 'SQLAdditions' panel with the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000.

Sol) **select name from city where CountryCode='USA' and population>120000;**

The screenshot shows the MySQL Workbench interface with a red border around the main window. The SQL editor tab contains the following code:

```

14 (3973,'Concord','USA','california',121780),
15 (3977,'Cedar Rapids','USA','Iowa',120758),
16 (3982,'Coral Springs','USA','Florida',117549),
17 (4054,'Farfield','USA','California',92256),
18 (4058,'Boulder','USA','Colorado',91238),
19 (4061,'Fall River','USA','Massachusetts',90555);

20

21 • select * from city;
22 • select * from city where CountryCode='USA' and population>100000;
23 • select name from city where CountryCode='USA' and population>120000;

```

The Result Grid shows the names of cities that meet the criteria:

name
Scottsdale
Corona
Concord
Cedar Rapids

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
7	08:52:03	select * from city where CountryCode='USA' and population>100000	5 row(s) returned	0.000 sec / 0.000 sec
8	09:20:37	select * from city	9 row(s) returned	0.016 sec / 0.000 sec
9	09:23:35	select name from city where CountryCode='USA' and population>120000	4 row(s) returned	0.000 sec / 0.000 sec

Q3. Query all columns (attributes) for every row in the CITY table.

Sol) **Select * from City;**

The screenshot shows the MySQL Workbench interface with a red border around the main window. The SQL editor tab contains the following code:

```

19 (4061,'Fall River','USA','Massachusetts',90555);

20

21 • select * from city where CountryCode='USA' and population>100000;
22 • select name from city where CountryCode='USA' and population>120000;
23 • select * from city;

```

The Result Grid shows all columns for each city row:

ID	Name	CountryCode	District	population
6	Rotterdam	NLD	Zuid-Holland	593321
3878	Scottsdale	USA	Arizona	202705
3965	Corona	USA	California	124966
3973	Concord	USA	California	121780
3977	Cedar Rapids	USA	Iowa	120758
3982	Coral Springs	USA	Florida	117549
4054	Farfield	USA	California	92256
4058	Boulder	USA	Colorado	91238
4061	Fall River	USA	Massachusetts	90555

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
8	09:20:37	select * from city	9 row(s) returned	0.016 sec / 0.000 sec
9	09:23:35	select name from city where CountryCode='USA' and population>120000	4 row(s) returned	0.000 sec / 0.000 sec
10	09:30:04	select * from city	9 row(s) returned	0.015 sec / 0.000 sec

Q4) Query all columns for a city in CITY with the ID 1661.

Sol) **Select * from city where ID = 1661;**

The screenshot shows the MySQL Workbench interface. In the central pane, there is a SQL editor tab titled "Sql_Window_Functions" containing the following SQL code:

```

20
21 • select * from city where CountryCode='USA' and population>100000;
22 • select name from city where CountryCode='USA' and population>120000;
23 • select * from city;
24 • select * from city where ID=1661;

```

Below the SQL editor is a "Result Grid" showing the following data:

ID	Name	CountryCode	District	population

The status bar at the bottom right indicates "21°C Sunny" and the system date/time.

Output - No rows returned.

Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

Sol) select * from city where CountryCode='JPN';

The screenshot shows the MySQL Workbench interface. In the central pane, there is a SQL editor tab titled "Sql_Window_Functions" containing the following SQL code:

```

16 (3982,'Coral Springs','USA','Florida',117549),
17 (4054,'Farfield','USA','California',92256),
18 (4058,'Boulder','USA','Colorado',91238),
19 (4061,'Fall River','USA','Massachusetts',90555);
20
21 • select * from city where CountryCode='USA' and population>100000;
22 • select name from city where CountryCode='USA' and population>120000;
23 • select * from city;
24 • select * from city where ID=1661;
25 • select * from city where CountryCode='JPN';

```

Below the SQL editor is a "Result Grid" showing the following data:

ID	Name	CountryCode	District	population

The status bar at the bottom right indicates "21°C Sunny" and the system date/time.

Output - No rows returned.

Q6) Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

Sol) Select name from City where CountryCode='JPN';

Output - No rows returned.

Sample Dataset 2

First of all , I created a new table ‘station’ and then inserted the records in it using multi-insert statement. The glimpse of it is shown in the image.

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, titled 'SQL File 4', there is a code block containing a CREATE TABLE statement and a multi-insert statement. The code is as follows:

```
28 • create table station(
29     ID int(10),
30     city varchar(17),
31     State varchar(3),
32     Lat_N int(10),
33     Long_W int(10));
34
35 • insert into station values(794,'Kissee Mills','MO',139,73),(824,'Loma Mar','CA',48,130),
36   (503,'Sandy Hook','CT',73,148),(478,'Tipton','IN',33,97),
37   (619,'Arlington','CO',75,92),(711,'Turner','AR',56,101),
38   (839,'Slideell','LA',85,151),(411,'Negreet','LA',98,105),
39   (588,'Glencoe','KY',46,136),(665,'Chelsea','IA',98,59),
40   (342,'Chignik Lagoon','AK',103,153),(733,'Pelahatchie','MS',38,28),
41   (441,'Hanna City','IL',59,136),(811,'Dorrance','KS',102,121),
42   (698,'Albany','CA',49,80),(325,'Monument','KS',70,141),
43   (414,'Manchester','MD',73,37),(113,'Prescott','IA',39,65),
44   (973,'Graettinger','IA',94,150),(266,'Cahone','CO',116,127);
45
46 • select * from station;
47 • select distinct(city) from station where MOD(ID,2)=0
48 order by city;
```

The 'Output' pane shows the results of the last two queries. The first query inserts 20 rows. The second query selects 10 distinct city names. The third query orders the results by city name.

Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

Sol) **Select distinct(city) from station where MOD(ID,2)=0
order by city;**

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, the same query as in the previous screenshot is run. The output shows the results of the query:

```
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
city
Albany
Cahone
Chignik Lagoon
Glencoe
Kissee Mills
Loma Mar
Manchester
Tipton
```

The 'Output' pane shows the execution details and the results of the query.

Q9) Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

Sol) **select (count(*) - count(distinct city)) as Difference
from station;**

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a query:

```

42 (698, 'Albany', 'CA', 49, 80), (325, 'Monument', 'KS', 78, 141),
43 (414, 'Manchester', 'ND', 73, 37), (113, 'Prescott', 'IA', 39, 65),
44 (971, 'Graettinger', 'IA', 94, 150), (266, 'Cahone', 'CO', 116, 127),
45
46 • select * from station;
47 • select distinct(city) from station where MOD(ID,2)=0
48 order by city;
49 • select (count(*) - count(distinct city)) as Difference
      from station;
50
51

```

The Result Grid shows the output of the last query:

Difference
0

The Action Output panel shows the execution log:

Action	Time	Message	Duration / Fetch
select * from station	25 10:15:38	20 row(s) returned	0.000 sec / 0.000 sec
select distinct(city) from station	26 10:16:08	20 row(s) returned	0.000 sec / 0.000 sec
select (count(*) - count(distinct city)) as Difference from station	27 10:16:26	1 row(s) returned	0.000 sec / 0.000 sec

There is zero difference between total no of cities and distinct cities. All the city records in the station table are unique.

Q10) Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

Sol) **(Select CITY,LENGTH(CITY) as Min from STATION
order by Length(CITY) asc, CITY limit 1)
union**

**(Select CITY,LENGTH(CITY) as Max from STATION order by
Length(CITY) desc, CITY limit 1);**

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a query:

```

51
52 • (select CITY,LENGTH(CITY) as Min from STATION order by Length(CITY) asc, CITY limit 1) union
53 (select CITY,LENGTH(CITY) as Max from STATION order by Length(CITY) desc, CITY limit 1);
54
55

```

The Result Grid shows the output of the query:

CITY	Min
Albany	6
Chignik Lagoon	14

Q11) Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

Sol) **Select Distinct(city) from station
where city like 'a%' OR CITY LIKE 'e%' OR CITY LIKE 'i%'
OR CITY LIKE 'o%' OR CITY LIKE 'u%';**

The screenshot shows the Oracle SQL Developer interface. The code editor contains the following SQL query:

```

55
56 • select Distinct(city) from station
57 where city like '%a' OR CITY LIKE '%e' OR CITY LIKE '%i%
58 OR CITY LIKE '%o' OR CITY LIKE '%u'

```

The results grid shows two rows: 'Arlington' and 'Albany'. The output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	10:47:48	select Distinct(city)from station where city like '%a' OR CITY LIKE '%e' OR CITY LIKE '%i% OR CITY LIKE '%o' OR CITY LIKE '%u';	2 row(s) returned	0.000 sec / 0.000 sec

Q12) Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

**Sol) Select Distinct(city) from station
where city like '%a%' OR CITY LIKE '%e%' OR CITY LIKE '%i%'
OR CITY LIKE '%o%' OR CITY LIKE '%u%';**

The screenshot shows the Oracle SQL Developer interface. The code editor contains the following SQL query:

```

59
60 • Select Distinct(city) from station
61 where city like '%a%' OR CITY LIKE '%e%' OR CITY LIKE '%i%
62 OR CITY LIKE '%o%' OR CITY LIKE '%u%'

```

The results grid shows five rows: 'Glencoe', 'Chelsea', 'Pelahatchie', 'Dorrance', and 'Cahone'. The output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	10:47:48	Select Distinct(city)from station where city like '%a%' OR CITY LIKE '%e%' OR CITY LIKE '%i% OR CITY LIKE '%o%' OR CITY LIKE '%u%';	2 row(s) returned	0.000 sec / 0.000 sec
2	10:51:20	Select Distinct(city)from station where city like "%a%" OR CITY LIKE "%e%" OR CITY LIKE "%i%" OR CITY LIKE "%o%" OR CITY LIKE "%u%";	5 row(s) returned	0.000 sec / 0.000 sec

Q13) Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

Select Distinct(city) from station where city not like 'a%' and CITY not LIKE 'e%' and CITY not LIKE 'i%' and CITY not LIKE 'o%' and CITY not LIKE 'u%';

The screenshot shows the Oracle SQL Developer interface. The code editor contains the following SQL query:

```

64 • Select Distinct(city) from station
65 where city not like 'a%' and CITY not LIKE 'e%' and CITY not LIKE '%i%
66 and CITY not LIKE '%o%' and CITY not LIKE '%u%'

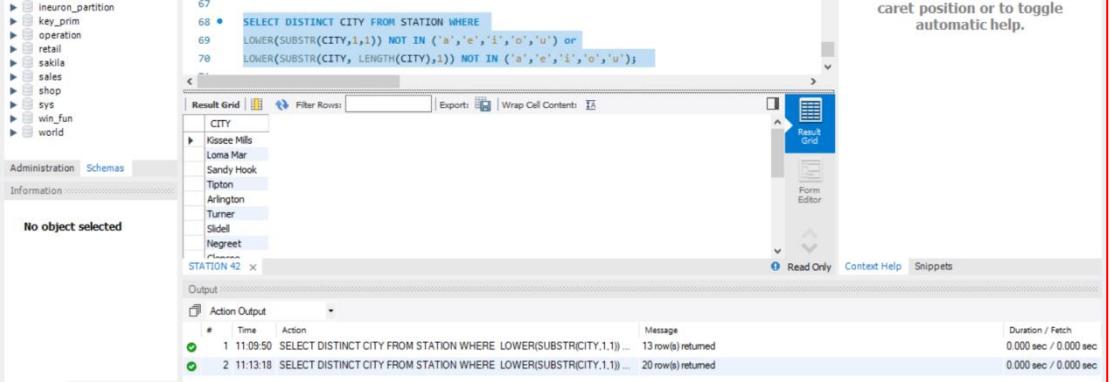
```

The results grid shows ten rows: 'Kissimmee', 'Loma Mar', 'Sandy Hook', 'Tipton', 'Turner', 'Slidell', 'Negreet', 'Glencoe', 'Chelsea', and 'Ognik Lagoon'. The output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
2	10:51:20	Select Distinct(city)from station where city not like 'a%' OR CITY not LIKE 'e%' OR CITY not LIKE 'i%' OR CITY not LIKE 'o%' OR CITY not LIKE 'u%';	5 row(s) returned	0.000 sec / 0.000 sec
3	10:54:43	Select Distinct(city)from station where city not like 'a%' OR CITY not LIKE 'e%' OR CITY not LIKE 'i%' OR CITY not LIKE 'o%' OR CITY not LIKE 'u%';	20 row(s) returned	0.000 sec / 0.000 sec
4	10:56:53	Select Distinct(city)from station where city not like 'a%' and CITY not LIKE 'e%' and CITY not LIKE 'i%' and CITY not LIKE 'o%' and ...	18 row(s) returned	0.000 sec / 0.000 sec

Q15) Query the list of CITY names from STATION that do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

**Sol) Select distinct city from station where
 lower(substr(city,1,1)) not in ('a','e','i','o','u')
 or
 lower(substr(city, length(city),1)) not in ('a','e','i','o','u');**



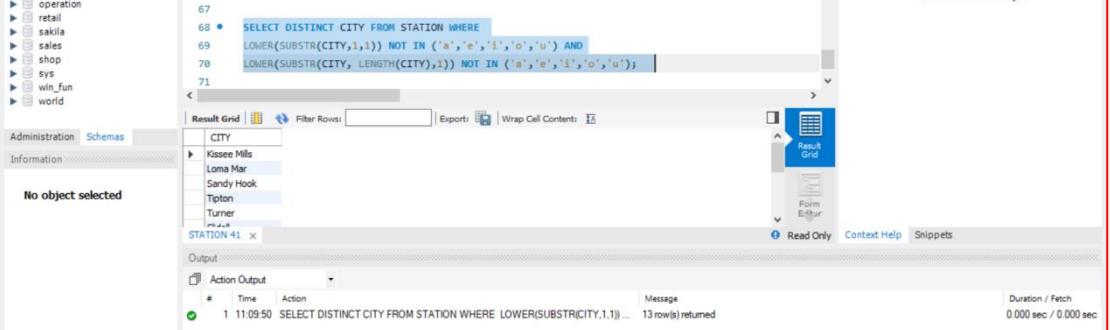
```

67
68 • SELECT DISTINCT CITY FROM STATION WHERE
69 LOWER(SUBSTR(CITY,1,1)) NOT IN ('a','e','i','o','u') OR
70 LOWER(SUBSTR(CITY, LENGTH(CITY),1)) NOT IN ('a','e','i','o','u')
    
```

The screenshot shows the MySQL Workbench interface. The left sidebar lists databases like `information_schema`, `key_prm`, `retail`, `sakila`, `sales`, `shop`, `sys`, `win_fun`, and `world`. The main area shows the results of the query, which returns 20 distinct city names: Kissimmee, Loma, Mar, Sandy, Hook, Tipton, Arlington, Turner, Slidell, Negreet, Clemons, and STATION 42.

Q16) Query the list of CITY names from STATION that do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

**Sol) Select distinct city from station where
 lower(substr(city,1,1)) not in ('a','e','i','o','u') and
 lower(substr(city, length(city),1)) not in ('a','e','i','o','u');**



```

67
68 • SELECT DISTINCT CITY FROM STATION WHERE
69 LOWER(SUBSTR(CITY,1,1)) NOT IN ('a','e','i','o','u') AND
70 LOWER(SUBSTR(CITY, LENGTH(CITY),1)) NOT IN ('a','e','i','o','u')
    
```

The screenshot shows the MySQL Workbench interface. The left sidebar lists databases like `information_schema`, `key_prm`, `retail`, `sakila`, `sales`, `shop`, `sys`, `win_fun`, and `world`. The main area shows the results of the query, which returns 13 distinct city names: Kissimmee, Loma, Mar, Sandy, Hook, Tipton, Arlington, Turner, Slidell, Negreet, Clemons, and STATION 41.

Q17) Table: Product

Column Name	Type
Product_id	int
Product_name	varchar
Unit_price	int

product_id is the primary key of this table.
 Each row of this table indicates the name and the price of each product

Table: Sales

Column Name	Type
Seller_id	int
Product_id	int
Buyer_id	int
Sales_date	date

Quantity	int
Price	int

This table has no primary key, it can have repeated rows. product_id is a foreign key to the Product table. Each row of this table contains some information about one sale.

Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.

First of all , I have created both tables Product and Sales and then inserted the records manually using insert command. The glimpse of it is shown below.

```

File Edit View Query Database Server Tools Scripting Help
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas Sql_Windows_Functions Assignment(SET!) Primary&Foreign
SCHEMAS
q Filter objects
dress_data
ineuron
ineuron_1
ineuron_partition
key_prim
operation
retail
sakila
sales
shop
sys
win_fun
world
Administration Schemas Information
72 • create table Product(
73     product_id int() not null primary key,
74     product_name varchar(10),
75     unit_price int(10));
76
77 • create table sales(
78     seller_id int(),
79     product_id int(),
80     buyer_id int(),
81     sales_date date,
82     quantity int(),
83     price int(),
84     foreign key(product_id) references Product(product_id));
85
86
87
88
89
90
91
92
93
94
95
96
97

```

```

key_prim
operation
retail
sakila
sales
shop
sys
win_fun
world
Administration Schemas Information No object selected
85
86 • insert into product values(1,'50',1000),
87     (3,'64',800),
88     (3,'Iphone',1400);
89
90 • select * from product;
91
92 • insert into sales values(1,1,'2019-01-21',2,2000),
93     (1,2,2,'2019-02-17',1,800),
94     (2,2,3,'2019-06-02',1,800),
95     (3,3,4,'2019-05-13',2,2000);
96
97 • select * from sales;

```

automatic help.

Output

#	Time	Action	Message	Duration / Fetch
9	11:45:21	insert into sales values(1,1,'2019-01-21',2,2000),(1,2,2,'2019-02-17',1,800),(2,2,3,'2019-06-02',1,800)	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.015 sec
10	11:45:34	select * from sales	4 row(s) returned	0.000 sec / 0.000 sec
11	11:45:49	use assignment	0 row(s) affected	0.016 sec

Sol) Select distinct product_id,product_name from product where product_id in (select product_id1 from sales where sales_date between '2019-01-01' AND '2019-03-31');

```

102 • insert into sales values(1,1,'2019-01-21',2,2800),
103 (1,2,2,'2019-02-17',1,800),
104 (2,3,3,'2019-06-02',1,800),
105 (3,3,4,'2019-05-13',2,2800);

106
107 • select * from sales;
108 • select distinct product_id,product_name from product
109 where product_id in (select product_id from sales
110 where sales_date between '2019-01-01' AND '2019-03-31');

111
112
113
114

```

Q18) Table Views:

Column_Name	Type
Article_id	int
Author_id	int
Viewer_id	int
View_date	date

Write an SQL query to find all the authors that viewed at least one of their own articles. Return the result table sorted by id in ascending order. Return the resulted table in ascending order.

Sol) First of all , I created a table ‘views’ and then inserted the records manually using insert function.

```

101
102 • create table views(
103     article_id int(5),
104     author_id int(5),
105     viewer_id int(5),
106     view_date date);
107
108 • insert into views values(1,3,5,'2019-08-01'),
109 (1,3,6,'2019-08-02'),
110 (2,7,7,'2019-08-01'),
111 (2,7,6,'2019-08-02'),
112 (4,7,1,'2019-07-22'),
113 (3,4,4,'2019-07-21'),
114 (3,4,4,'2019-07-21');

```

Input:

Article_id	Author_id	Viewer_id	View_date
1	3	5	2019-08-01
1	3	6	2019-08-02
2	7	7	2019-08-01
2	7	6	2019-08-02
4	7	1	2019-07-22
3	4	4	2019-07-21
3	4	4	2019-07-21

Output:

id
4
7

```
select distinct(author_id) as id from views  
where author_id in (select viewer_id from views  
where author_id=viewer_id)  
Order by author_id;
```

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view is open, showing various databases and tables like 'classicmodels', 'neuron', 'neuron_1', etc. In the center, the SQL editor contains the provided SQL query. Below it, the 'Result Grid' shows the output with two rows: 'id' (4) and 'id' (7). To the right of the grid is a 'Result Grid' button. At the bottom, the 'Action Output' pane displays log entries for the executed statements, including the select query, a use assignment, and the final select query which returned 2 rows.

Q19) The table holds information about food delivery to customers that make orders at some date and specify a preferred delivery date (on the same order date or after it). If the customer's preferred delivery date is the same as the order date, then the order is called immediately; otherwise, it is called scheduled. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

Sol) First of all , I created a table ‘delivery’ and then inserted the records manually using insert function.

The screenshot shows the MySQL Workbench interface. The 'Schemas' tree view is open. The SQL editor contains the code to create the 'delivery' table and insert data into it. The table has columns: delivery_id (primary key), customer_id, order_date (date type), and customer_pref_delivery_date (date type). Six rows of data are inserted, showing various combinations of dates. Below the editor, the 'Output' pane shows the results of the 'select' statement and the warning message about integer display width being deprecated.

with cte as (select * from
(select delivery_id, customer_id, order_date, customer_pref_delivery_date,
(Case when order_date=customer_pref_delivery_date then 1 else 0 end) as
immediate,

```

RANK() OVER(PARTITION BY customer_id ORDER BY order_date) as
first_order
from delivery)x)
SELECT
round(sum(immediate) *100 / count(first_order),2) as immediate_percentage
from cte;

```

The screenshot shows a database interface with a sidebar containing various schemas like assignment, dress_data, imuron, imuron_1, imuron_partition, key_prim, operation, retail, and sakila. The main area displays the SQL query. A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The result grid shows a single row with the value 33.33. The status bar at the bottom shows the execution time as 0.000 sec / 0.000 sec.

Q20) A company is running Ads and wants to calculate the performance of each Ad. Performance of the Ad is measured using Click-Through Rate (CTR) where: Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points. Return the result table ordered by ctr in descending order and by ad_id in ascending order in case of a tie.

Input:

Column	Data type
Ad_id	int
User_id	int
Action	enum

Output:

Ad_id	CTR
1	66.67
3	50
2	33.33
5	0

First of all , I have created a table ‘ads’ and then inserted the data manually using multi insert function.

Here, We have to Calculate Click Through Rate(CTR) which is used to calculate performance of the ads. The formula to calculate it is:

$$\text{CTR} = \left\{ 0, \frac{\text{Ad(Total clicks)}}{\text{Ad(total clicks)} + \text{Ad(Total views)}} \right\}$$

Note- Ignored ads is not considered.

```

142
143 • create table ads(
144     ad_id int,
145     user_id int,
146     act_ion varchar(15),
147     primary key(ad_id,user_id));
148
149 • insert into ads values(1,1,'clicked'),
150     (2,2,'clicked'),
151     (3,3,'viewed'),
152     (5,5,'ignored'),
153     (1,7,'ignored'),
154     (2,7,'viewed'),
155     (3,5,'clicked'),
156     (1,4,'viewed'),
157     (2,11,'viewed'),
158     (1,2,'clicked');
159

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

Select distinct ad_id, ifnull(
    round(sum(act_ion = 'Clicked') / (sum(act_ion = 'Clicked') + sum(act_ion = 'Viewed')) * 100, 2),0)
as ctr
from ads
group by ad_id
order by ctr desc, ad_id;

```

ad_id	ctr
1	66.67
3	50.00
2	33.33
5	0.00

disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Q21) Table: Employee

Column Name	Type
Employee_id	int
Team_id	int

Employee_id is the primary key for this table. Each row of this table contains the ID of each employee and their respective team. Write an SQL query to find the team size of each of the employees. Return result table in any order. The query result format is in the following example.

Output:

Employee_id	Team_size
-------------	-----------

Sol) As similar to above procedures, I have created an empty table first named ‘employee’ and then added records in it. And then performed the desired query over it. Here we have to find out team size of each employee of the table.

```

169 • create table employee(
170   emp_id int(5) primary key,
171   team_id int(5));
172
173 • insert into employee values(1,8),
174   (2,8),(3,8),(4,7),(5,9),(6,9);
175
176

```

Action Output

#	Time	Action	Message	Duration / Fetch
3	07:57:45	show tables	7 row(s) returned	0.000 sec / 0.000 sec
4	07:58:35	create table employee(emp_id int(5) primary key, team_id int(5))	0 row(s) affected, 2 warning(s): 1601 Integer display width is deprecated and will be ...	0.078 sec
5	07:59:52	insert into employee values(1,8), (2,8),(3,8),(4,7),(5,9),(6,9)	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.016 sec

**Select emp_id,
count(team_id) over (partition by team_id) as team_size
from employee
order by emp_id;**

```

175
176 • select emp_id,
177   count(team_id) over (partition by team_id) as team_size
178   from employee
179   order by emp_id;

```

Result Grid

emp_id	team_size
1	3
2	3
3	3
4	1
5	2
6	2

Action Output

#	Time	Action	Message	Duration / Fetch
1	08:08:14	select emp_id, count(team_id) over (partition by team_id) as team_size from employee	6 row(s) returned	0.000 sec / 0.000 sec
2	08:08:43	select emp_id, count(team_id) over (partition by team_id) as team_size from employee...	6 row(s) returned	0.015 sec / 0.000 sec

Q22) Write an SQL query to find the type of weather in each country for November 2019.

The type of weather is:

- Cold if the average weather_state is less than or equal 15,

-

- Hot if the average weather_state is greater than or equal to 25, and

- Warm otherwise.

Return result table in any order

Input: Countries table

Column Name	Type
Country_id	int
Country_name	varchar

Weather Table:

Column Name	Type
-------------	------

Country_id	int
Weather state	int
day	date

Output:

Country Name	Weather Type
USA	Cold
Australia	Cold
Peru	Hot
Morocco	Hot
China	Warm

Sol) At first, I created two empty table named ‘countries’ and ‘weather’ and then inserted the records row by row. Here we have to find out the Country name and Weather type of every country.

```

assignment
Tables
Views
Stored Procedures
Functions
dress_data
neuron
neuron_1
neuron_partition
key_prim
operation
detail
sakila
sales
Administration Schemas
Information
No object selected

181
182 • Create table countries(
183     country_id int primary key,
184     country_name varchar(10));
185
186 • create table weather(
187     country_id int,
188     weather_state int,
189     day date,
190     primary key(country_id,day));
191
192 • insert into countries values(2,'USA'),
193     (3,'Australia'),(5,'Peru'),(6,'Morocco'),(9,'Spain');
194
195 • insert into weather values(2,15,'2019-11-01'),
196     (2,12,'2019-10-28'),(2,12,'2019-10-27'),(3,8,'2019-11-10'),
197     (3,3,'2019-11-12'),(5,16,'2019-11-07'),(5,18,'2019-11-09'),
198     (5,23,'2019-11-23'),(7,25,'2019-11-28'),(7,22,'2019-12-01'),
199     (7,20,'2019-12-02'),(8,25,'2019-11-05'),
     (8,27,'2019-11-15'),(8,31,'2019-11-25'),(9,7,'2019-11-23'),
     (9,3,'2019-12-23');

Automatic context help is
disabled. Use the toolbar to
manually get help for the current
caret position or to toggle
automatic help.

Context Help Snippets
Output

```

```

select country_name,
(case when avg(weather_state)<=15 then 'cold'
      when avg(weather_state)>=25 then 'hot'
      else 'warm'
    end) as 'weather_type'
from countries c inner join weather w
on c.country_id = w.country_id
where w.day between '2019-11-01' and '2019-11-30'
group by c.country_id;

```

```

assignment
Tables
Views
Stored Procedures
Functions
dress_data
neuron
neuron_1
neuron_partition
key_prim
operation
detail
sakila
sales
Administration Schemas
Information
No object selected

200
201 • select country_name,
202     (case when avg(weather_state)<=15 then 'cold'
203           when avg(weather_state)>=25 then 'hot'
204           else 'warm'
205         end) as 'weather_type'
206     from countries c inner join weather w
207     on c.country_id = w.country_id
208     where w.day between '2019-11-01' and '2019-11-30'
209     group by c.country_id;

Automatic context help is
disabled. Use the toolbar to
manually get help for the current
caret position or to toggle
automatic help.

Result Grid | Filter Rows: Export: Wrap Cell Content: 
country_name weather_type
USA cold
Australia cold
China warm
Peru hot
Morocco hot
Spain cold

Result 9 ×
Read Only Context Help Snippets
Output
Action Output
# Time Action
Object Info Session
8 08:27:50 insert into weather values(2,15,2019-11-01), (2,12,2019-10-28),(2,12,2019-10-27) ... 17 row(s) affected Records: 17 Duplicates: 0 Warnings: 0 Duration / Fetch 0.015 sec

```

**Q23) Write an SQL query to find the average selling price for each product.
average_price should be
rounded to 2 decimal places.
Return the result table in any order**

Input:

Prices Table

Column Name	Type
Product_id	int
Start_date	date
End_date	date
price	int

UnitsSold Table

Column Name	Type
Product_id	int
purchase_date	date
Units	int

Output:

Product_id	Average_price
1	6.96
2	16.96

Sol) Initially,I created two empty tables named ‘Price’ and ‘UnitsSold’
And then inserted records in it. And then performed the desired query.

```

211 • Ⓜ create table Prices(
212   product_id int,
213   start_date date,
214   end_date date,
215   price int,
216   primary key(product_id,start_date,end_date));
217
218 • Ⓜ create table UnitsSold(
219   product_id int,
220   purchase_date date,
221   units int);
222
223 • insert into Prices values(1,'2019-02-17','2019-02-28',5),
224   (1,'2019-03-01','2019-03-22',20),
225   (3,'2019-02-01','2019-02-28',15),
226   (2,'2019-02-21','2019-03-31',30);
227
228 • insert into UnitsSold values(1,'2019-02-25',100),
229   (1,'2019-03-01',15),(1,'2019-02-10',200),(1,'2019-03-22',30);

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

select u.product_id,
round(sum(p.price * u.units)/sum(u.units),2) as average_price
from prices p left join UnitsSold u
on p.product_id = u.product_id and (u.purchase_date between p.start_date
And p.end_date) group by u.product_id;

```

```

drop table UnitsSold;
select u.product_id,
       round(sum(p.price * u.units)/sum(u.units),2) as average_price
  from prices p left join UnitsSold u
    on p.product_id = u.product_id
   and (u.purchase_date between p.start_date And p.end_date)
 group by u.product_id;

```

product_id	average_price
1	6.96
2	16.96

Result 17 x

Action Output

#	Time	Action	Message	Duration / Fetch
23	09:28:30	drop table UnitsSold	0 row(s) affected	0.032 sec
24	09:28:57	create table UnitsSold(product_id int, purchase_date date, units int)	0 row(s) affected	0.046 sec
25	09:29:03	insert into UnitsSold values(1,2019-02-25,100), (1,2019-03-01,15),(2,2019-02-10,...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
26	09:29:10	select u.product_id,round(sum(p.price * u.units)/sum(u.units),2) as average_price fr...	2 row(s) returned	0.000 sec / 0.000 sec

Q24) Input: Activity Table

Column Name	Type
Player_id	int
Device_id	int
Event_date	date
Games_played	int

(player_id, event_date) is the primary key of this table.
This table shows the activity of players of some games.
Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on someday using some device.

Write an SQL query to report the first login date for each player.

Output:

Player_id	First_login
1	2016-03-01
2	2017-06-25
3	2016-03-02

Sol) Initially, I created a empty table ‘Activity’ and then loaded the data inside it.

```

create table Activity(
  player_id int,
  device_id int,
  event_date date,
  games_played int,
  primary key(player_id,event_date));

```

```

insert into Activity values(1,2,'2016-03-01',5),
(1,2,'2016-05-02',6),
(1,3,'2017-06-25',1),
(3,3,'2017-06-25',1),
(3,1,'2016-03-02',0),
(3,4,'2018-07-03',5);

```

#	Time	Action	Message	Duration / Fetch
25	09:29:03	insert into UnitsSold values(1,2019-02-25,100), (1,2019-03-01,15),(2,2019-02-10,...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
26	09:29:10	select u.product_id,round(sum(p.price * u.units)/sum(u.units),2) as average_price fr...	2 row(s) returned	0.000 sec / 0.000 sec
27	09:43:33	create table Activity(player_id int, device_id int, event_date date, games_played int,...	0 row(s) affected	0.046 sec
28	09:46:06	insert into Activity values(1,2,2016-03-01,5), (1,2,2016-05-02,6), (2,3,2017-06-25,...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec

And then performed the desired query over it.

**select player_id,MIN(event_date) as first_login
from activity**

group by player_id;

```

252
253 • select player_id,MIN(event_date) as first_login
254   from activity
255   group by player_id;
  
```

The screenshot shows the MySQL Workbench interface. The left sidebar lists databases: dress_data, inuron, inuron_1, inuron_partition, key_prim, operation, retail, sakila, sales, and chinook. The central pane displays the SQL query and its results. The results table has two columns: player_id and first_login. The data shows three rows: player_id 1 with first_login 2016-03-01, player_id 2 with first_login 2017-06-25, and player_id 3 with first_login 2016-03-02.

player_id	first_login
1	2016-03-01
2	2017-06-25
3	2016-03-02

The bottom pane shows the Action Output log with four entries:

- 26 09:29:10 select u.product_id,round((sum(p.price * u.units)/sum(u.units)),2) as average_price fr... 2 row(s) returned 0.000 sec / 0.000 sec
- 27 09:43:33 create table Activity(player_id int, device_id int, event_date date, games_played int, ... 0 row(s) affected 0.046 sec
- 28 09:46:06 insert into Activity values(1,2,2016-03-01,5),(1,2,2016-05-02,6),(2,3,2017-06-25,... 5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0 0.016 sec
- 29 10:49:42 select player_id,MIN(event_date) as first_login from activity group by player_id 3 row(s) returned 0.000 sec / 0.000 sec

Q25) Write a SQL query to report the device that is first logged in for each player. Return the table in any order.

Output:

Player_id	Device_id
1	2
2	3
3	1

Sol) **select distinct(player_id),
FIRST_VALUE(device_id) OVER (PARTITION BY player_id ORDER BY
event_date ASC) AS device_id
from activity;**

```

252
253 • select distinct(player_id),
254   FIRST_VALUE(device_id) OVER (PARTITION BY player_id ORDER BY event_date ASC) AS device_id
255   from activity;
  
```

The screenshot shows the MySQL Workbench interface. The left sidebar lists databases: dress_data, inuron, inuron_1, inuron_partition, key_prim, operation, retail, sakila, sales, and chinook. The central pane displays the SQL query and its results. The results table has two columns: player_id and device_id. The data shows three rows: player_id 1 with device_id 2, player_id 2 with device_id 3, and player_id 3 with device_id 1.

player_id	device_id
1	2
2	3
3	1

The bottom pane shows the Action Output log with one entry:

- 1 10:59:44 select distinct(player_id), FIRST_VALUE(device_id) OVER (PARTITION BY player_id... 3 row(s) returned 0.000 sec / 0.000 sec

**Q26) Input:
Products Table**

Column Name	Type
Product_id	int
Product_name	varchar
Product_category	varchar

product_id is the primary key for this table. This table contains data about the company's products.

Orders Table

Column Name	Type
Product_id	int
Order date	date
unit	int

There is no primary key for this table. It may have duplicate rows.

product_id is a foreign key to the Products table. unit is the number of products ordered in order_date.

Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount. Return result table in any order. The query result format is in the following example.

Output:

Product_name	Unit
Leetcode solutions	130
Leetcode kit	100

Sol) Create two empty table 'Products' & 'Orders' table and then inserted records in it row by row. And then perform the desired query.

```
Select p.product_name, sum(o.unit) as unit from products p
left join orders o
on p.product_id = o.product_id
where o.order_date Between '2020-02-01' And '2020-02-29'
group by p.product_name
having sum(o.unit)>=100;
```

The screenshot shows the following steps:

- Table Creation:**
 - Line 257: create table products(product_id int primary key, product_name varchar(30), product_category varchar(20));
 - Line 261: create table Orders(product_id int, order_date date, unit int, foreign key(product_id) references products(product_id));
- Data Insertion:**
 - Line 265: insert into products values(1,'Leetcode Solutions','Book'),(2,'Jewels of Stringology','Book'),(3,'HP','Laptop'),(4,'Lenovo','Laptop'),(5,'Leetcode Kit','Tshirt');
 - Line 268: insert into Orders values(1,'2020-02-05',60),(1,'2020-02-10',70),(2,'2020-01-18',30),(2,'2020-02-11',80),(3,'2020-02-17',2),(3,'2020-02-24',3),(4,'2020-03-01',20),(4,'2020-03-04',30),(4,'2020-03-04',60),(5,'2020-02-25',50),(5,'2020-02-27',50),(5,'2020-03-01',50);
- Execution and Results:**
 - Line 272: The query is executed, showing the results in the 'Action Output' pane.
 - The results are:
 - 1 11:21:08 Insert into Orders values(1,2020-02-05,60),(1,2020-02-10,70),(2,2020-01-18,30)... 12 row(s) affected Records: 12 Duplicates: 0 Warnings: 0 Duration / Fetch: 0.016 sec
 - 2 11:21:29 select * from Products 5 row(s) returned 0.000 sec / 0.000 sec
 - 3 11:21:36 select * from Orders 12 row(s) returned 0.000 sec / 0.000 sec
 - 4 11:21:51 use assignment 0 row(s) affected 0.000 sec

```

273 • Select p.product_name, sum(o.unit) as unit from products p
274   left join orders o
275   on p.product_id = o.product_id
276   where o.order_date Between '2020-02-01' And '2020-02-29'
277   group by p.product_name
278   having sum(o.unit)>=100;

```

disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Q27) Input: Users Table

Column Name	Type
user_id	int
name	varchar
mail	varchar

user_id is the primary key for this table.

This table contains information of the users signed up in a website. Some emails are invalid.

Write an SQL query to find the users who have valid emails.

A valid e-mail has a prefix name and a domain where:

- The prefix name is a string that may contain letters (upper or lower case), digits, underscore

'_', period '.', and/or dash '-'. The prefix name must start with a letter.

- The domain is '@leetcode.com'.

Return the result table in any order.

The query result format is in the following example

Sol) Create an empty table ‘Users’ and then inserted the records manually row by row.

Then performed the desired query.

```

277   group by p.product_name
278   having sum(o.unit)>=100;
279
280 • create table Users(
281   user_id int primary key,
282   name varchar(30),
283   mail varchar(50));
284
285 • insert into Users values(1,'Winston','winston@leetcode.com'),
286   (2,'Jonathan','jonathanisgreat'),
287   (3,'Annabelle','bella@leetcode.com'),
288   (4,'Sally','sally.com@leetcode.com'),
289   (5,'Marwan','quartz#2020@leetcode.com'),
290   (6,'David','davidis@gmai.com'),
291   (7,'Shapiro','.shapo@leetcode.com');

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

select *
from users
where regexp_like(mail, '^[a-zA-Z][a-zA-Z0-9_\\.-]*@leetcode.com');

```

user_id	name	mail
1	Winston	winston@leetcode.com
3	Annabelle	bella@leetcode.com
4	Sally	sally.com@leetcode.com
NULL	NULL	NULL

Action Output

#	Time	Action	Message	Duration / Fetch
1	11:42:58	Select p.product_name, sum(o.quantity) as unit from products p left join orders o on p.product_id = o.product_id group by p.product_id;	2 row(s) returned	0.000 sec / 0.000 sec
2	11:48:36	create table Users(user_id int primary key, name varchar(30), mail varchar(50))	0 row(s) affected	0.094 sec
3	11:52:16	insert into Users values(1,'Winston','winston@leetcode.com'),(2,'Jonathan','jonathan@leetcode.com'),(3,'Sally','sally.com@leetcode.com')	7 row(s) affected Records: 7 Duplicates: 0 Warnings: 0	0.016 sec
4	11:52:55	SELECT * FROM Users WHERE REGEXP_LIKE(mail, '^*[a-zA-Z][a-zA-Z0-9_\\.-]*@leetcode.com')	3 row(s) returned	0.016 sec / 0.000 sec

Q28) Write an SQL query to report the customer_id and customer_name of customers who have spent at least \$100 in each month of June and July 2020. Return the result table in any order.

The query result format is in the following example

Sol) First of all , I have created 3 empty tables namely ‘Customers’,’Produkt’ and ‘Orders’. Then manually inserted records in it row by row and then performed the query. Here all the three tables are connected to each other, through some column.

We have to use it as a condition and find the desired output.

```

create table customers(
customer_id int primary key, name varchar(20), country varchar(20));
create table produkt(product_id int primary key, description varchar(20), price int);
create table Orders(order_id int primary key, customer_id int, product_id int, order_date date, quantity int);
insert into customers values(1,'Winston','USA'),(2,'Jonathan','Peru'),(3,'Mousafa','Egypt');
insert into produkt values(10,'LC Phone',300),(20,'LC Tshirt',10),(30,'LC Book',45),(40,'LC Key Chain',5);
insert into Orders values(1,1,10,'2020-06-10',1),(2,1,20,'2020-07-01',1),
(3,1,30,'2020-07-08',2),(4,2,10,'2020-06-15',2),(5,2,40,'2020-07-01',10),
(6,3,20,'2020-06-24',2),(7,3,30,'2020-06-25',2),(8,3,30,'2020-05-08',3);

```

```

select o.customer_id, c.name
from Customers c, Produkt p, Orders o
where c.customer_id = o.customer_id and p.product_id = o.product_id
group by o.customer_id
having (
    sum(case when o.order_date like '2020-06%' then o.quantity*p.price else 0 end) >= 100
)
and
    sum(case when o.order_date like '2020-07%' then o.quantity*p.price else 0 end) >= 100;

```

A screenshot of a database query editor. On the left, there's a tree view of tables: assignment, activity, ads, city, countries, delivery, employee, prices, product, sales, station, unitsold, users. The 'assignment' table is selected. The main area shows a SQL query:

```

327
328 • select o.customer_id, c.name
  from Customers c, Produkt p, Orders o
  where c.customer_id = o.customer_id and p.product_id = o.product_id
  group by o.customer_id
329
330
331
332
333
334
335

```

The query includes two HAVING clauses with SUM(CASE) conditions. Below the query is a result grid:

customer_id	name
1	Winston

At the bottom right, there are buttons for 'Result Grid', 'Form Editor', 'Read Only', 'Context Help', and 'Snippets'. A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Q29) Input:

Table: TVProgram

Column Name	Type
program_date	date
content_id	int
channel	varchar

Table: Content

Column Name	Type
content_id	varchar
title	varchar
Kids_content	enum
content_type	varchar

Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020. Return the result table in any order.
The query result format is in the following example.

Sol) Create an empty table 'TvProgram' and 'Content' then inserted the records manually row by row. Then performed the desired query.

A screenshot of a database query editor. On the left, there's a tree view of tables: activity, ads, city, countries, delivery, employee, prices, product, sales, station, unitsold, users. The 'activity' table is selected. The main area shows a SQL query:

```

323 • create table Tvprogram(program_date date,
324   content_id int,channel varchar(15),primary key(program_date,content_id));
325
326 • create table Content(content_id varchar(20) primary key,
327   title varchar(20),kids_content varchar(15),content_type varchar(15));
328
329 • insert into Tvprogram values('2020-06-10 08:00',1,'LC Channel'),
330   ('2020-05-11 12:00',2,'LC Channel'),('2020-05-12 12:00',3,'LC Channel'),
331   ('2020-05-13 14:00',4,'Disney Ch'),('2020-06-18 14:00',4,'Disney Ch'),
332   ('2020-07-15 16:00',5,'Disney Ch');
333
334 • insert into content values(1,'Leetcode Movie','N','Movies'),
335   (2,'Alg. for kids','Y','Series'),(3,'Database Sols','N','Series'),
336   (4,'Aladdin','Y','Movies'),(5,'Cinderella','Y','Movies');

```

Below the query is an 'Output' section with an 'Action Output' table:

#	Time	Action	Message	Duration / Fetch
10	16:15:14	create table Content(content_id varchar(20) primary key, title varchar(20),kids_content varchar(15),content_type varchar(15))	0 row(s) affected	0.047 sec
11	16:20:09	insert into Tvprogram values('2020-06-10 08:00',1,LC Channel)	(2020-05-11 12:00,... 6 row(s) affected, 6 warning(s): 1292 Incorrect date value: '2020-06-10 08:00' for co... 0.015 sec	
12	16:22:26	insert into content values(1,'Leetcode Movie','N','Movies'),(2,'Alg. for kids','Y','Series'),(3,'Database Sols','N','Series'),(4,'Aladdin','Y','Movies'),(5,'Cinderella','Y','Movies')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0 0.031 sec	

At the bottom right, there are buttons for 'Context Help' and 'Snippets'. A tooltip on the right says: "manually get help for the current caret position or to toggle automatic help."

Select distinct(title) from content c
join Tvprogram t on c.content_id=t.content_id
where c.kids_content = 'Y' and c.content_type = 'Movies'

and (month(program_date), year(program_date)) = (6, 2020);

The screenshot shows a MySQL Workbench session. In the top pane, there is a code editor with the following SQL query:

```
337
338 • select distinct(title) from content c
339 join Tvpromgram t on c.content_id=t.content_id
340 where c.kids_content = 'Y' and c.content_type = 'Movies'
341 and (month(program_date), year(program_date)) = (6, 2020);
```

The results pane below shows a single row in a grid:

title
Aladdin

Below the results, the "Output" tab displays the execution log:

#	Time	Action	Message	Duration / Fetch
11	16:20:09	insert into Tvpromgram values('2020-06-10 08:00:00','LC Channel'), ('2020-05-11 12:00:00','Cartoon'), ('2020-05-11 12:00:00','Cartoon'), ('2020-05-11 12:00:00','Cartoon'), ('2020-05-11 12:00:00','Cartoon'), ('2020-05-11 12:00:00','Cartoon')	6 row(s) affected, 6 warning(s); 1292 Incorrect date value: '2020-06-10 08:00' for column 'program_date';	0.015 sec
12	16:22:26	insert into content values(1,'Leetcode Movie','N','Movies'), (2,'Alg for kids','Y','Serie')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
13	16:32:01	select distinct(title) from content c join Tvpromgram t on c.content_id=t.content_id where c.kids_content = 'Y' and c.content_type = 'Movies' and (month(program_date), year(program_date)) = (6, 2020);	1 row(s) returned	0.016 sec / 0.000

Q30) Table: NPV

Column Name	Type
id	int
year	int
npv	int

(id, year) is the primary key of this table.

Table: Queries

Column Name	Type
id	int
year	int

Write an SQL query to find the npv of each query of the Queries table.

Return the result table in any order. The query result format is in the following example.

Sol) Create two empty table 'NPV' and 'Queries' then inserted the records manually row by row.Then performed the desired query.

The screenshot shows the SSMS interface with a query window titled "Assignment(SET1)*". The code in the window is:

```

338 • select distinct(title) from content c
join Typrogram t on c.content_id=t.content_id
where c.kids_content = 'Y' and c.content_type = 'Movies'
and (month(program_date), year(program_date)) = (8, 2020);

343 • Create table NPV(
    id int,year int,npv int,primary key(id,year));
345
346 • Create table Queries(
    id int,year int, primary key(id,year));
348
349 • insert into NPV values(1,2018,100),
(7,2020,30),(13,2019,40),(1,2019,113),(2,2008,121),
(3,2009,12),(11,2028,98),(7,2015,0)
352
353 • insert into Queries values(1,2019),(2,2008),(3,2009),(7,2018),
(7,2019),(7,2020),(13,2019);
354

```

The output pane shows the results of the last three statements:

Action	Time	Action	Message	Duration / Fetch
15	16:40:56	create table Queries(id int,year int, primary key(id,year))	0 row(s) affected	0.047 sec
16	16:42:58	insert into NPV values(1,2018,100),(7,2020,30),(13,2019,40),(1,2019,113),(2,2008,...	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.032 sec
17	16:44:23	insert into Queries values(1,2019),(2,2008),(3,2009),(7,2018),(7,2019),(7,2020),(13,...	7 row(s) affected Records: 7 Duplicates: 0 Warnings: 0	0.016 sec

Select q.id,q.year,n.npv from queries q left join NPV n
on q.id = n.id and n.year = q.year;

The screenshot shows the SSMS interface with a query window containing the following SQL statement:

```

356 • select q.id,q.year,n.npv from queries q left join NPV n
on q.id = n.id and n.year = q.year;

```

The result grid displays the following data:

	id	year	npv
1	2019	113	
2	2008	121	
3	2009	12	
7	2018	0	
7	2019	0	
7	2020	30	
13	2019	40	

Q31) SAME QUESTION AS Q30

Q32) Table: Employees

Column Name	Type
id	int
name	varchar

id is the primary key for this table.

Each row of this table contains the id and the name of an employee in a company.

Table: EmployeeUNI

Column Name	Type
id	int
unique_id	int

(id, unique_id) is the primary key for this table.

Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.

Sol) Create two empty table ‘Employees’ and ‘EmployeesUNI’ then inserted the records manually row by row. Then performed the desired query.

```

358 • create table employees(
359   id int primary key, name varchar(20));
360
361
362 • create table employeesUNI(
363   id int,unique_id int,primary key(id,unique_id));
364
365 • insert into employees values(1,'Alice'),(7,'Bob'),(11,'Meir'),
366   (90,'Winston'),(3,'Jonathan');
367
368 • insert into employeesUNI values(3,1),(11,2),(90,3);
  
```

Action Output:

#	Time	Action	Message	Duration / Fetch
26	17:00:12	create table employeesUNI(id int,unique_id int,primary key(id,unique_id))	Error Code: 1050. Table 'employeesuni' already exists	0.000 sec
27	17:02:03	insert into employees values(1,'Alice'),(7,'Bob'),(11,'Meir'),(90,'Winston'),(3,'Jonathan')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
28	17:03:05	insert into employeesUNI values(3,1),(11,2),(90,3)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.015 sec

In this query, I have used Left join function because it is asked to show null values if a user does not have unique id.

```

SELECT b.unique_id, a.name
FROM employees a
LEFT JOIN employeesUNI b on a.id = b.id;
  
```

```

369
370 • SELECT b.unique_id, a.name
371   FROM employees a
372   LEFT JOIN employeesUNI b on a.id = b.id;
  
```

Result Grid:

unique_id	name
NULL	Alice
1	Jonathan
2	Bob
3	Meir
3	Winston

Action Output:

#	Time	Action	Message	Duration / Fetch
1	17:06:47	SELECT b.unique_id, a.name FROM employees a LEFT JOIN employeesUNI b on a.i... returned	5 row(s) returned	0.000 sec / 0.000 sec

Q33) Table: Users

Column Name	Type
id	int
name	varchar

id is the primary key for this table. name is the name of the user.

Table: Rides

Column Name	Type
id	int
user_id	int
distance	int

id is the primary key for this table.

user_id is the id of the user who travelled the distance "distance".

Write an SQL query to report the distance travelled by each user.

Return the result table ordered by travelled_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order. The query result format is in the following example

Sol) Create two empty table ‘Users’ and ‘Rides’ then inserted the records manually row by row. Then performed the desired query.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', there are several tables listed: queries, product, sales, station, tvprogram, unitssold, views, and weather. Below these, 'Views' and 'Stored Procedures' are shown. The main area displays the following SQL code:

```
374 • create table users(id int primary key,name varchar(15));
375
376 • create table Rides(
377     id int primary key,
378     user_id int,distance int);
379
380 • insert into Users values(1,'Alice'),(2,'Bob'),
381     (3,'Alex'),(4,'Donald'),(7,'lee'),(13,'Jonathan'),(19,'Elvis');
382
383 • insert into Rides values(1,1,120),(2,2,317),(3,3,222),
384     (4,7,100),(5,13,312),(6,19,50),(7,7,120),(8,19,400),(9,7,230);
385
```

Below the code, the 'Output' tab shows the execution results:

#	Time	Action	Message	Duration / Fetch
11	18:50:03	select u.name,sum(ifnull(r.distance,0)) as travelled_distance from users u inner join...	6 row(s) returned	0.000 sec / 0.000 sec
12	18:50:27	select u.name,sum(ifnull(r.distance,0)) as travelled_distance from users u left join rd...	7 row(s) returned	0.016 sec / 0.000 sec
13	18:51:06	use assignment	0 row(s) affected	0.000 sec

```
select u.name,sum(ifnull(r.distance,0)) as travelled_distance
from users u
left join rides r
on u.id = r.user_id
group by u.name
order by travelled_distance desc,u.name asc;
```

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', there are several tables listed: employee, orders, npv, prices, product, queries, product, sales, station, tvprogram, unitssold, views, and weather. Below these, 'Views' and 'Stored Procedures' are shown. The main area displays the following SQL code:

```
385
386 • select u.name,sum(ifnull(r.distance,0)) as travelled_distance
387     from users u
388     left join rides r
389     on u.id = r.user_id
390     group by u.name
391     order by travelled_distance desc,u.name asc;
```

The results are displayed in a grid:

name	travelled_distance
Elvis	450
lee	450
Bob	317
Jonathan	312
Alex	222
Alice	120
Donald	0

Below the grid, the 'Output' tab shows the execution results:

#	Time	Action	Message	Duration / Fetch
10	18:46:56	select u.name,sum(ifnull(r.distance,0)) as travelled_distance from users u inner join rides r o...	6 row(s) returned	0.000 sec / 0.000 sec
11	18:50:03	select u.name,sum(ifnull(r.distance,0)) as travelled_distance from users u inner join...	6 row(s) returned	0.000 sec / 0.000 sec
12	18:50:27	select u.name,sum(ifnull(r.distance,0)) as travelled_distance from users u left join rd...	7 row(s) returned	0.016 sec / 0.000 sec

Q34) SAME QUESTION AS Q26

Q35) Table: Movies

Column Name	Type
movie_id	int
title	varchar

Table: Users

Column Name	Type
user_id	int
name	varchar

Table: MovieRating

Column Name	Type
movie_id	int
user_id	int
rating	int
created_at	date

Write an SQL query to:

- Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name.
- Find the movie name with the highest average rating in February 2020. In case of a tie, return the lexicographically smaller movie name.

Sol) Create three empty table ‘Movies’, ‘Users’ and ‘Movierating’ then inserted the records manually row by row. Then performed the desired query.

```

392
393 • create table movies(movie_id int primary key,title varchar(15));
394
395 • create table users(user_id int primary key,name varchar(15));
396
397 • create table movierating(movie_id int,user_id int,
398 rating int,created_at date,
399 primary key(movie_id,user_id));
400
401 • insert into movies values(1,'Avengers'),(2,'Frozen 2'),(3,'Joker');
402
403 • insert into users values(1,'Daniel'),(2,'Monica'),(3,'Maria'),(4,'James');
404
405 • insert into movierating values(1,1,3,'2020-01-12'),
406 (1,2,4,'2020-02-11'),(1,3,2,'2020-02-12'),(1,4,1,'2020-01-01'),
407 (2,1,5,'2020-02-17'),(2,2,2,'2020-02-01'),(2,3,2,'2020-03-01'),
408 (3,1,3,'2020-02-22'),(3,2,4,'2020-02-25');

#  Action Output
#  Time      Action
18 19:02:13  insert into movies values(1,'Avengers'),(2,'Frozen 2'),(3,'Joker') 3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0 0.015 sec
19 19:03:07  insert into users values(1,'Daniel'),(2,'Monica'),(3,'Maria'),(4,'James') 4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0 0.016 sec
20 19:06:28  insert into movierating values(1,1,3,'2020-01-12'),(1,2,4,'2020-02-11'),(1,3,2,'2020-02-12'),(1,4,1,'2020-01-01'),(2,1,5,'2020-02-17'),(2,2,2,'2020-02-01'),(2,3,2,'2020-03-01'),(3,1,3,'2020-02-22'),(3,2,4,'2020-02-25') 9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0 0.031 sec

```

(select b.name as results from users b
join movierating c on b.user_id=c.user_id
group by b.user_id

```

order by count(*) desc,b.name asc limit 1)
UNION
(select a.title as results from movies a
join movierating c on a.movie_id=c.movie_id
where (month(c.created_at), year(c.created_at)) = (2, 2020)
group by c.movie_id
order by avg(c.rating) desc,a.title asc limit 1);

```

The screenshot shows a database interface with a sidebar containing various schemas like 'content', 'countries', 'customers', etc. The main area displays the SQL query. A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The results pane shows two rows: 'Daniel' and 'Frozen 2'.

Q36) SAME QUESTION AS Q33

Q37) SAME QUESTION AS Q32

Q38) Table: Departments

Column Name	Type
id	int
name	varchar

id is the primary key of this table. The table has information about the id of each department of a university.

Table: Students

Column Name	Type
id	int
name	varchar
department_id	int

id is the primary key of this table. The table has information about the id of each student at a university and the id of the department he/she studies at.

Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist. Return the result table in any order.

Sol) Create two empty table ‘Departments’ and ‘Student’ then inserted the records manually row by row. Then performed the desired query.

The screenshot shows the SQL Server Management Studio interface. In the top navigation bar, there are tabs like 'Sql_Windows_functions', 'Assignment(SET!)', 'Sql_Joins_unions_CTE', 'Primary&Foreign', and 'SQLAdditions'. The left sidebar shows 'SCHEMAS' with various tables listed: delivery, employee, employeesuniv, orders, npn, prices, product, series, product, rides, sales, station, typrogram. The main pane displays the following SQL code:

```

420 •  create table department(id int primary key,name varchar(25));
421 •  create table student(id int primary key,name varchar(15),dep_id int);
422
423 •  insert into department values(1,'Electrical engineering');
424     ,(7,'Computer engineering'),(13,'Business Administration');
425
426
427 •  insert into student values(23,'Alice',1),(1,'Bob',7),
428     (5,'Jennifer',13),(2,'John',14),(4,'Jasmine',77),(3,'Steve',74),
429     (6,'Luis',1),(8,'Jonathan',7),(7,'Dianna',33),(11,'Madelynn',1);
430

```

The results grid shows the following data:

ID	Name
2	John
3	Steve
4	Jasmine
7	Dianna

The output pane shows the execution log:

- # 16 19:59:51 select s.id,s.name from student s left join department d on d.id = s.dep_id where s.id... 0 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec
- # 17 20:00:15 select s.id,s.name from student s left join department d on d.id = s.dep_id where d.id... 4 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec

**select s.id,s.name from student s
left join department d on d.id = s.dep_id
where d.id is null;**

The screenshot shows the SQL Server Management Studio interface. The code pane contains the same SQL query as above:

```

431
432 •  select s.id,s.name from student s
433     left join department d on d.id = s.dep_id
434     where d.id is null;

```

The results grid shows the following data:

ID	Name
2	John
3	Steve
4	Jasmine
7	Dianna

The output pane shows the execution log:

- # 16 19:59:51 select s.id,s.name from student s left join department d on d.id = s.dep_id where s.id... 0 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec
- # 17 20:00:15 select s.id,s.name from student s left join department d on d.id = s.dep_id where d.id... 4 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec

Q39) Table: Calls

Column Name	Type
from_id	int
to_id	int
duration	int

This table does not have a primary key, it may contain duplicates.
This table contains the duration of a phone call between from_id and to_id. from_id != to_id

Write an SQL query to report the number of calls and the total call duration between each pair of distinct persons (person1, person2) where person1 < person2.

Sol) Create an empty table ‘Calls’ then inserted the records manually row by row. Then performed the desired query.

```

434
435 • 435 create table calls(from_id int,
436   to_id int, duration int);
437
438 • 438 insert into calls values (1,2,50), (2,1,11),
439   (1,3,20), (3,4,100),(3,4,200),(3,4,200),(4,3,499);
440
441 • 441 select from_id as person1,to_id as person2,
442   count(duration) as call_count,sum(duration) as total_duration
443
444
445
446
447
448
449
  
```

The screenshot shows the SQL editor with the following code:

```

create table calls(from_id int,
to_id int, duration int);

insert into calls values (1,2,50), (2,1,11),
(1,3,20), (3,4,100),(3,4,200),(3,4,200),(4,3,499);

select from_id as person1,to_id as person2,
count(duration) as call_count,sum(duration) as total_duration
from (select * from calls)
  
```

Select from_id as person1,to_id as person2,
Count(duration) as call_count,sum(duration) as total_calls
From (select * from calls)

Union all

Select to_id,from_id,duration
From calls) as a

Where from_id < to_id

Group by person1,person2;

The screenshot shows the SQL editor with the following code:

```

select from_id as person1,to_id as person2,
count(duration) as call_count,sum(duration) as total_duration
from (select * from Calls
union all
select to_id, from_id, duration
from Calls) as a
where from_id < to_id
group by person1,person2;
  
```

The Result Grid shows the following data:

person1	person2	call_count	total_duration
1	2	2	70
1	3	1	20
3	4	4	999

The Output pane shows the following log:

```

Result 28 x
Output
Action Output
# Time Action Message Duration / Fetch
22 20:20:26 select from_id as person1,to_id as person2, count(duration) as call_count,sum(dur... Error Code: 1248. Every derived table must have its own alias 0.000 sec
23 20:20:57 select from_id as person1,to_id as person2, count(duration) as call_count,sum(dur... 3 row(s) returned 0.000 sec / 0.000 sec
  
```

Q40) SAME QUESTION AS Q23

Q41.

Table: Warehouse

Column Name	Type
name	varchar
product_id	int
units	int

(name, product_id) is the primary key for this table. Each row of this table contains the information of the products in each warehouse.

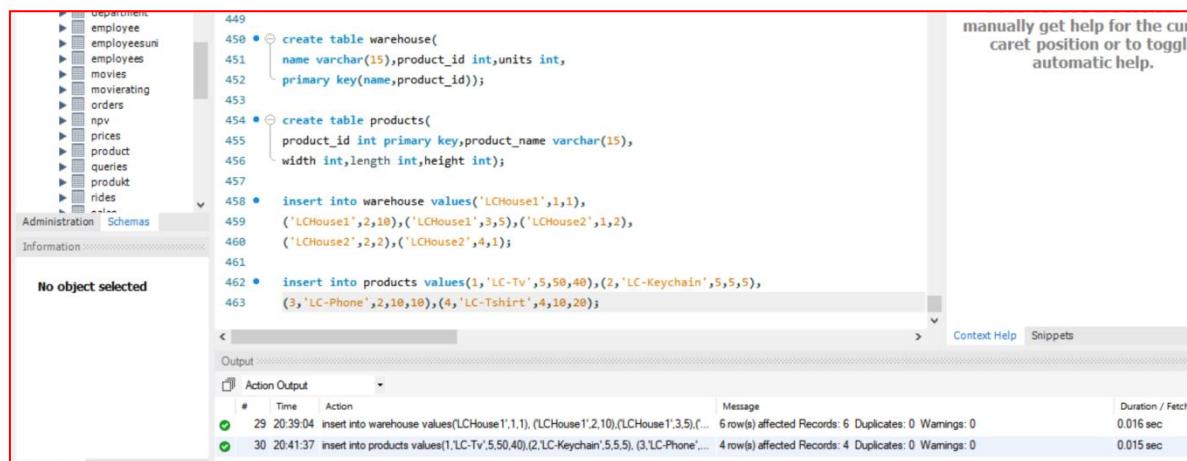
Table: Products

Column Name	Type
product_id	int
product_name	varchar
Width	int
Length	int
Height	int

product_id is the primary key for this table. Each row of this table contains information about the product dimensions (Width, Length, and Height) in feet of each product.

Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse. Return the result table in any order. The query result format is in the following example.

Sol) Create two empty table ‘Warehouse’ and ‘Products’ then inserted the records manually row by row. Then performed the desired query.



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

- Schemas:** department, employee, employeesuni, employees, movies, moderating, orders, npv, prices, product, queries, product, rides, sales.
- Current Schema:** Administration
- Code Editor:**

```

449
450 •  create table warehouse(
451     name varchar(15),product_id int,units int,
452     primary key(name,product_id));
453
454 •  create table products(
455     product_id int primary key,product_name varchar(15),
456     width int,length int,height int);
457
458 •  insert into warehouse values('LCHouse1',1,1),
459 ('LCHouse1',2,10),('LCHouse1',3,5),('LCHouse2',1,2),
460 ('LCHouse2',2,2),('LCHouse2',4,1);
461
462 •  insert into products values(1,'LC-TV',5,50,40),(2,'LC-Keychain',5,5,5),
463 (3,'LC-Phone',2,10,10),(4,'LC-Tshirt',4,10,20)

```
- Output:**

Action	Time	Action	Message	Duration / Fetch
29	20:39:04	insert into warehouse values(LCHouse1',1,1), (LCHouse1',2,10),(LCHouse1',3,5),(...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.016 sec
30	20:41:37	insert into products values(1,LC-TV',5,50,40),(2,LC-Keychain',5,5,5), (3,LC-Phone',...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.015 sec

```

select name as warehouse_name,
       sum(units*width*length*height) as volume
  from warehouse w
 inner join products p on w.product_id = p.product_id
 group by name
 order by null;

```

The screenshot shows the MySQL Workbench interface. On the left, there's a tree view of databases and tables, with 'movies' selected. In the center, the SQL editor contains the provided query. Below it, the 'Result Grid' shows the output:

warehouse_name	volume
LCHouse1	12250
LCHouse2	21050

At the bottom, the 'Output' pane displays the execution log:

```

Action Output
# Time Action Message Duration / Fetch
1 7 20:56:38 select name as warehouse_name, sum(units * vol) as volume from Warehouse w rig... 2 row(s) returned 0.016 sec / 0.000 sec
2 8 20:56:50 select name as warehouse_name, sum(units * vol) as volume from Warehouse w in... 2 row(s) returned 0.016 sec / 0.000 sec

```

Q42) Table: Sales

Column Name	Type
sale_date	date
fruit	enum
sold_num	int

(sale_date, fruit) is the primary key for this table.

This table contains the sales of "apples" and "oranges" sold each day.

Write an SQL query to report the difference between the number of apples and oranges sold each day. Return the result table ordered by sale_date.

The query result format is in the following example

Sol) create an empty table 'sales' and insert records inside it row by row manually. And then perform the desired query in it.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Administration' and 'Schemas', there are several tables listed: prices, product, products, queries, product, rides, station, movies, movierating, movies, orders, npv, prices, product, products, queries, product, rides, station. The 'movies' table is selected, and its structure is shown in the center pane:

```

Table: movies
Columns:
  movie_id int PK
  title     varchar(15)

```

In the SQL editor at the bottom, the following code is written:

```

478
479 • create table sales(
480   sale_date date,
481   fruit varchar(10),sold_num int,
482   primary key(sale_date,fruit));
483
484 • insert into sales values('2020-05-01','apples',10),
485   ('2020-05-01','oranges',8),('2020-05-02','apples',15),
486   ('2020-05-02','oranges',15),('2020-05-03','apples',20),
487   ('2020-05-03','oranges',0),('2020-05-04','apples',15),
488   ('2020-05-04','oranges',10);
489

```

The 'Output' pane shows the results of the last two statements:

#	Time	Action	Message	Duration / Fetch
11	22:22:48	create table sales(sale_date date, fruit varchar(10),sold_num int, primary key(sale_date,fruit))	0 rows affected	0.016 sec
12	22:25:28	insert into sales values('2020-05-01','apples',10),('2020-05-01','oranges',8),('2020-05-02','apples',15),('2020-05-02','oranges',15),('2020-05-03','apples',20),('2020-05-03','oranges',0),('2020-05-04','apples',15),('2020-05-04','oranges',10)	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.016 sec

```

select sale_date,
SUM(case when fruit='apples' then sold_num
else -sold_num
end) as diff
from sales
group by sale_date order by sale_date;

```

The screenshot shows the MySQL Workbench interface. The 'movies' table is selected in the schema list. The SQL editor contains the same query as above:

```

489
490 • select sale_date,
491   SUM(case when fruit='apples' then sold_num
492     else -sold_num
493   end) as diff
494   from sales
495   group by sale_date order by sale_date;

```

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

sale_date	diff
2020-05-01	2
2020-05-02	0
2020-05-03	20
2020-05-04	-1

The 'Output' pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
14	22:31:14	select sale_date, (case when fruit='apples' then sold_num else -sold_num end) as d... as diff from sales group by sale_date order by sale_date;	4 row(s) returned	0.016 sec / 0.000 sec
15	22:31:47	select sale_date, SUM(case when fruit='apples' then sold_num else -sold_num end) as diff from sales group by sale_date order by sale_date;	4 row(s) returned	0.000 sec / 0.000 sec

Q43) Table: Activity

Column Name	Type
player_id	int
device_id	int
event_date	date
games_played	int

(player_id, event_date) is the primary key of this table. This table shows the activity of players of some games. Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on someday using some device.

Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of play.

Sol) create an empty table ‘Activity’ and insert records inside it row by row manually. And then perform the desired query in it.

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree shows the 'movies' table under the 'Information' schema. The 'Tables' section also lists 'employees', 'employeesur', and 'employeesun'. The central pane displays the following SQL code:

```

496
497 • create table activity(
498     player_id int, device_id int, event_date date,
499     games_played int, primary key(player_id, event_date));
500
501 • insert into activity values(1,2,'2016-03-01',5),
502     (1,2,'2016-03-02',6),(2,3,'2017-06-25',1),
503     (3,1,'2016-03-02',0),(3,4,'2018-07-03',5);
504

```

The 'Output' pane shows the results of the 'create table' and 'insert' statements. The 'Message' section indicates 5 rows affected for the insert statement. The 'Duration / Fetch' section shows 0.016 sec for the create table and 0.031 sec for the insert.

```

with activity_stats as (
    select player_id, event_date,
        datediff(event_date,min(event_date) over(partition by player_id)) as date_diff
    from activity)
select round(count(distinct player_id) / (select count(distinct player_id)
    from activity), 2) as fraction
from activity_stats
where date_diff = 1;

```

The screenshot shows the MySQL Workbench interface. The 'Schemas' tree on the left shows the 'movies' table under the 'Information' schema. The central pane displays the previously written SQL query. A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The 'Result Grid' pane shows the result of the query: 'fraction' with a value of '0.33'. The 'Output' pane shows the execution details, including the insert into 'activity' and the execution of the query itself. The 'Message' section indicates 5 rows affected for the insert and 1 row(s) returned for the query. The 'Duration / Fetch' section shows 0.031 sec for the insert and 0.015 sec for the query.

Q44) Table: Employee

Column Name	Type
id	int
name	varchar
department	varchar
managerId	int

id is the primary key column for this table.

Each row of this table indicates the name of an employee, their department, and the id of their manager. If manager_Id is null, then the employee does not have a manager. No employee will be the manager of themselves.

Write an SQL query to report the managers with at least five direct reports. Return the result table in any order. The query result format is in the following example.

Sol) create an empty table ‘Employee’ and insert records inside it row by row manually. And then perform the desired query in it.

```
create table employee(
    id int primary key,
    name varchar(15),department varchar(15),managerid int);

insert into employee values(101,'John','A',Null),(102,'Dan','A',101),
(103,'James','A',101),(104,'Amy','A',101),
(105,'Anne','A',101),(106,'Ron','B',101);
```

The screenshot shows a database interface with a tree view on the left listing various tables like activity, ads, calls, city, content, countries, customers, delivery, department, employees, employeesun, and movingunit. The 'Information' tab is selected. In the center, a code editor window displays the creation of the 'employee' table and its initial data insertion. The table structure is defined with columns for id (primary key), name, and department. The data consists of six rows with IDs 101 through 106, names ranging from John to Ron, and departments A or B. Some rows have Null values for managerid. Below the code editor is a 'Result Grid' showing the inserted data. On the right, there's a context help panel and a OneDrive integration showing a screenshot was saved. The bottom status bar indicates 'No object selected'.

**select name from employee
where id in
(select managerid from employee
group by managerid
having count(managerid)>=5);**

Q45) Table: Student

Column Name	Type
student_id	int
student_name	varchar
gender	varchar
dept_id	int

student_id is the primary key column for this table. dept_id is a foreign key to dept_id in the Department tables. Each row of this table indicates the name of a student, their gender, and the id of their department.

Table: Department

Column Name	Type
dept_id	int
dept_name	varchar

dept_id is the primary key column for this table. Each row of this table contains the id and the name of a department.

Write an SQL query to report the respective department name and number of students majoring in each department for all departments in the Department table (even ones with no current students). Return the result table ordered by student_number in descending order. In case of a tie, order them by dept_name alphabetically. The query result format is in the following example.

Sol) create two empty table ‘Student’ & ‘Employee’ and insert records inside it row by row manually. And then perform the desired query in it.



```

527
528 • create table student(
529   student_id int primary key,
530   student_name varchar(15),gender varchar(15),depid int,
531   foreign key(depid) references department(dep_id));
532
533 • create table department(
534   dep_id int primary key,
535   dep_name varchar(15));
536
537 • insert into student values(1,'Jack','M',1),(2,'Jane','F',1),
538 (3,'Mark','M',2);
539
540 • insert into department values(1,'Engineering'),(2,'Science'),(3,'Law');
541

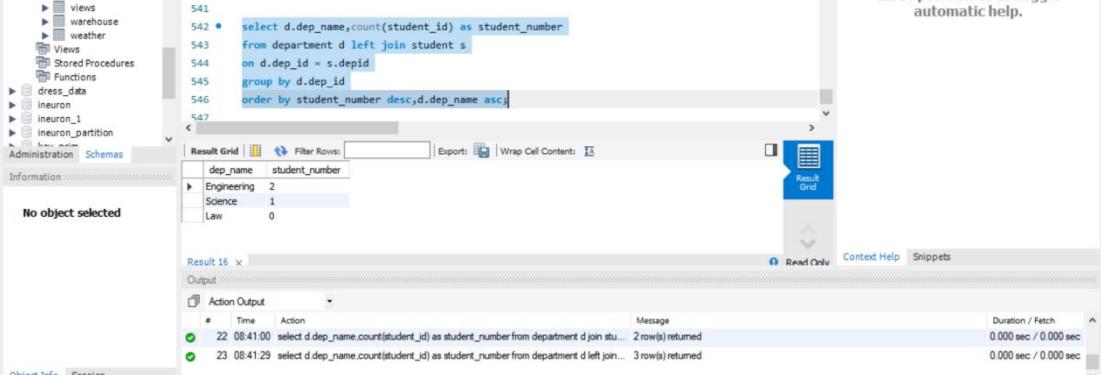
```

The screenshot shows the MySQL Workbench interface. On the left, the schema browser displays tables like `users`, `views`, `warehouse`, and `weather`. The central pane shows the SQL editor with the creation and insertion of data into the `student` and `department` tables. The output pane at the bottom shows the results of the `select` query from the previous step.

```

select d.dep_name,count(student_id) as student_number
from department d left join student s
on d.dep_id = s.depid
group by d.dep_id
order by student_number desc,d.dep_name asc;

```



```

541
542 • select d.dep_name,count(student_id) as student_number
543   from department d left join student s
544   on d.dep_id = s.depid
545   group by d.dep_id
546   order by student_number desc,d.dep_name asc;
547

```

dep_name	student_number
Engineering	2
Science	1
Law	0

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query for selecting data from the `department` and `student` tables. The results are displayed in a grid below, showing the count of students per department. The output pane at the bottom shows the execution details.

Q46)

Table: Customer

Column Name	Type
customer_id	int
product_key	int

There is no primary key for this table. It may contain duplicates. Product_key is a foreign key to the Product table.

Table: Product

Column Name	Type
product_key	int

Product_key is a primary key for this table.

Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table. Return the result table in any order. The query result format is in the following example.

Sol) create two empty table ‘Student’ & ‘Employee’ and insert records inside it row by row manually. And then perform the desired query in it.



```

547
548 • create table Product1(product_key int primary key);
549
550 • create table Customer1(customer_id int ,
551     product_key1 int,
552     foreign key(product_key1) references Product1(product_key));
553
554 • insert into Product1 values(5),(6);
555 • insert into Customer1 values(1,5),(2,6),(3,5),(3,6),(1,6);
556

```

disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

**select c.customer_id from customer1 c
group by customer_id
having count(distinct(product_key1)) in
(select count(product_key) from product1);**



```

557
558 • select c.customer_id from customer1 c
559 group by customer_id
560 having count(distinct(product_key1)) in
561 (select count(product_key) from product1);

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Read Only Context Help Snippets

No object selected

customer1 19 x Output

Action Output # Time Action Message Duration / Fetch

32 09:02:54 select c.customer_id from customer1 c where count(distinct(product_key1)) in ... Error Code: 1111. Invalid use of group function 0.000 sec

33 09:03:36 select c.customer_id from customer1 c group by customer_id having count(distinct(... 2 row(s) returned 0.016 sec / 0.000 sec

Q47) Table: Project

Column Name	Type
Project_id	int
Employee_id	int

(project_id, employee_id) is the primary key of this table. employee_id is a foreign key to the Employee table. Each row of this table indicates that the employee with employee_id is working on the project with project_id.

Table: Employee

Column Name	Type
employee_id	int
name	varchar
experience_years	int

employee_id is the primary key of this table. Each row of this table contains information about one employee.

Write an SQL query that reports the most experienced employees in each project. In case of a tie, report all employees with the maximum number of experience years. Return the result table in any order. The query result format is in the following example.

Sol) create two empty table ‘Project’ & ‘Employee’ and insert records inside it row by row manually. And then perform the desired query in it.

```

562 • create table employee1(employee_id int primary key, name varchar(15),
563     experience_years int);
564
565 • create table project(project_id int, employee_id1 int,
566     primary key(project_id, employee_id1),
567     foreign key(employee_id1) references employee1(employee_id));
568
569 • insert into employee1 values(1,'Khaled',3),
570     (2,'Ali',2),(3,'John',3),(4,'Doe',2);
571
572 • insert into project values(1,1),(1,2),(1,3),(2,1),(2,4);
573

```

```

Select project_id,employee_id1 FROM (
    SELECT p.project_id, p.employee_id1,
    DENSE_RANK() OVER(PARTITION BY p.project_id
ORDER BY e.experience_years DESC) as rnk
    FROM project p JOIN employee1 as e
    ON p.employee_id1 = e.employee_id
        ) as a
WHERE rnk = 1;

```

```

573
574 * Select project_id,employee_id1 FROM (
575     SELECT p.project_id, p.employee_id1,
576        DENSE_RANK() OVER(PARTITION BY p.project_id ORDER BY e.experience_years DESC) as rnk
577     FROM project p JOIN employeee1 as e
578     ON p.employee_id1 = e.employee_id
579 ) as a
580 WHERE rnk = 1;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Read Only | Context Help | Snippets

project_id	employee_id1
1	1
1	3
2	1

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:00:13	Select project_id,employee_id1 FROM (SELECT p.project_id,p.employee_id1,DEN...	3 row(s) returned	0.000 sec / 0.000 sec

Q48) Table: Books

Column Name	Type
book_id	int
name	varchar
available_from	date

book_id is the primary key of this table.

Table: Orders

Column Name	Type
order_id	int
book_id	int
quantity	int
dispatch_date	date

order_id is the primary key of this table. Book_id is a foreign key to the Books table.

Write an SQL query that reports the books that have sold less than 10 copies in the last year, excluding books that have been available for less than one month from today. Assume today is 2019-06-23.

Return the result table in any order. The query result format is in the following example.

.Sol) create two empty table ‘Books’ & ‘Orders’ and insert records inside it row by row manually. And then perform the desired query in it.

ORDER table missing in question.

Q49.

Table: Enrollments

Column Name	Type
student_id	int
course_id	int
grade	int

(student_id, course_id) is the primary key of this table.

Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course_id. Return the result table ordered by student_id in ascending order. The query result format is in the following example.

.Sol) create an empty table ‘Enrollments’ and insert records inside it row by row manually. And then perform the desired query in it.

The screenshot shows the MySQL Workbench interface. On the left, the 'Information' pane shows 'No object selected'. The main area displays the following SQL code:

```
598 ► project
599 ► queries
600 ► rides
601
602
603 ► create table enrollments (student_id int, course_id int,
604 | grade int,
605 | primary key(student_id, course_id));
606
607
608 ► insert into enrollments values(2,2,95),
609 | (2,3,95),(1,1,90),(1,2,99),(3,1,88),(3,2,75),
610 | (3,3,82);
611
612
613
614
615
616
```

The 'Output' pane shows the results of the insertions:

#	Time	Action	Message	Duration / Fetch
11	15:21:42	insert into books values('Kala & Demna','2010-01-01), ('2:28 letters','2012-05-10')	5 rows(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
12	15:26:55	create table enrollments (student_id int, course_id int, grade int, primary key(student_id, course_id))	0 rows(s) affected	0.031 sec
13	15:28:41	insert into enrollments values(2,2,95), (2,3,95),(1,1,90),(1,2,99),(3,1,88),(3,2,75), (3,3,82)	7 rows(s) affected Records: 7 Duplicates: 0 Warnings: 0	0.015 sec

```
select student_id, min(course_id) as course_id, grade
from Enrollments
where (student_id, grade) in
(select student_id, max(grade)
from Enrollments
group by student_id)
group by student_id
order by student_id asc;
```

```

employee1
employee
employees
employeesuni
movierating
movies
npv
prices
product
product1
products
product2
project
queries
rides

Administration Schemas
Information
No object selected

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
student_id course_id grade
1 2 99
2 2 95
3 3 82

Result 27 x
Output
Action Output
# Time Action Message Duration / Fetch
16 15:36:44 use assignment 0 row(s) affected 0.016 sec
17 15:37:27 select student_id, min(course_id) as course_id, grade from Enrollments where (student_id, grade) in (select student_id, max(grade) from Enrollments group by student_id) group by student_id order by student_id asc; 3 row(s) returned 0.000 sec / 0.000 sec

```

Q50)

Table: Players

Column Name	Type
player_id	int
Group_id	int

team_id is the primary key of this table. Each row of this table represents a single football team.

Table: Matches

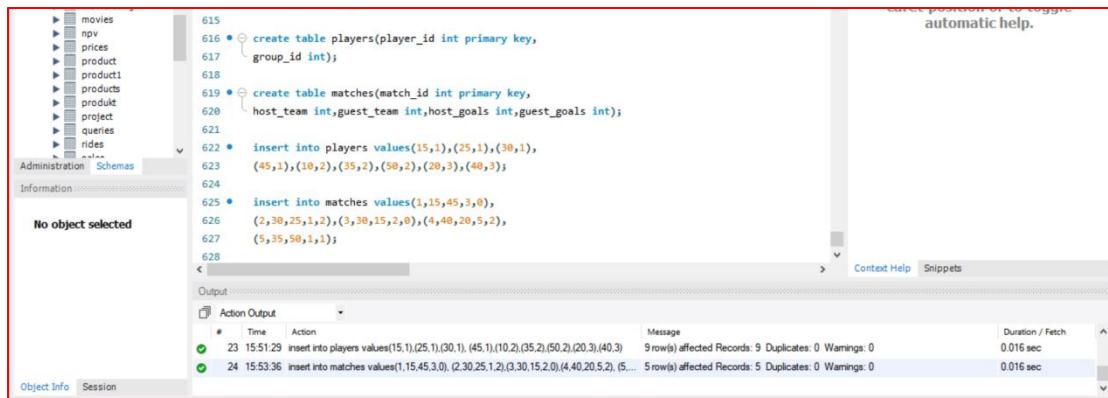
Column Name	Type
match_id	int
host_team	int
guest_team	int
host_goals	int
guest_goals	int

match_id is the primary key of this table.

Each row is a record of a finished match between two different teams. Teams host_team and guest_team are represented by their IDs in the Teams table (team_id), and they scored host_goals and guest_goals goals, respectively. The winner in each group is the player who scored the maximum total points within the group. In the case of a tie, the lowest player_id wins.

**Write an SQL query to find the winner in each group.
Return the result table in any order. The query result format is in the following example.**

.Sol) create two empty table ‘Players’ & Matches and insert records inside it row by row manually. And then perform the desired query in it.



```

615
616 • create table players(player_id int primary key,
617     group_id int);
618
619 • create table matches(match_id int primary key,
620     host_team int, guest_team int, host_goals int, guest_goals int);
621
622 • insert into players values(15,1),(25,1),(30,1),
623 (45,1),(10,2),(35,2),(50,2),(20,3),(40,3);
624
625 • insert into matches values(1,15,45,3,0),
626 (2,30,25,1,2),(3,30,15,2,0),(4,40,20,5,2),
627 (5,35,50,1,1);
628

```

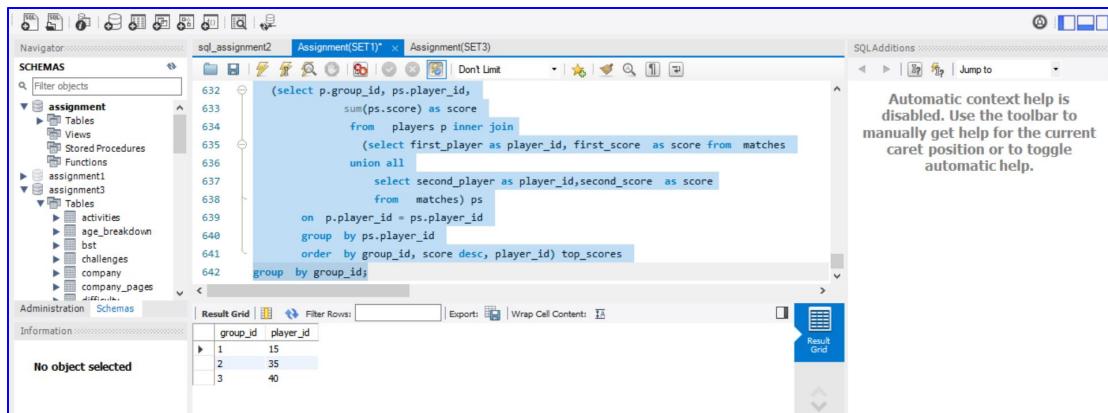
Action Output

#	Time	Action	Message	Duration / Fetch
23	15:51:29	insert into players values(15,1),(25,1),(30,1), (45,1),(10,2),(35,2),(50,2),(20,3),(40,3)	9 rows affected Records: 9 Duplicates: 0 Warnings: 0	0.016 sec
24	15:53:36	insert into matches values(1,15,45,3,0), (2,30,25,1,2),(3,30,15,2,0),(4,40,20,5,2), (5,35,50,1,1)	5 rows affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec

```

select group_id, player_id from
(select p.group_id, ps.player_id,
sum(ps.score) as score
from players p inner join
(select first_player as player_id, first_score as
score from matches
union all
select second_player as player_id,second_score as score
from matches) ps
on p.player_id = ps.player_id
group by ps.player_id
order by group_id, score desc, player_id) top_scores
group by group_id;

```



group_id	player_id
1	15
2	35
3	40