SQL Challenge

Manasa Panidapu

(21WU0101055)

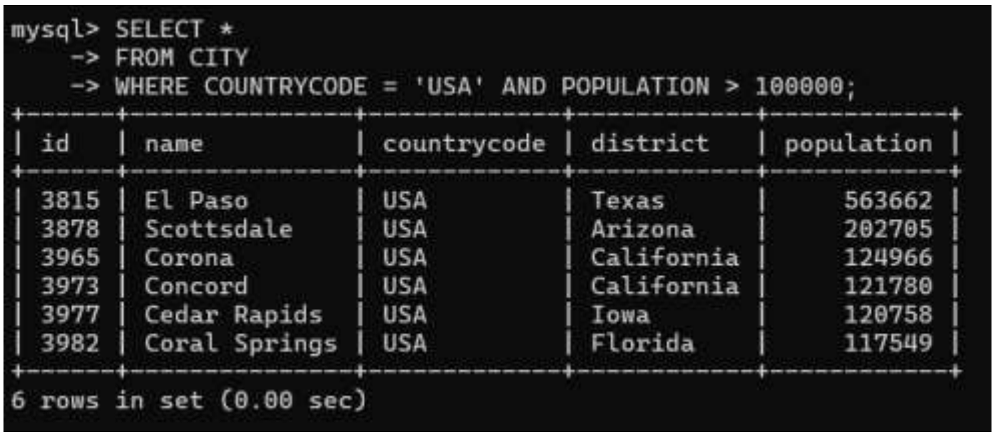
Team 3

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

SELECT *

FROM CITY

WHERE COUNTRYCODE = 'USA' AND POPULATION > 100000;



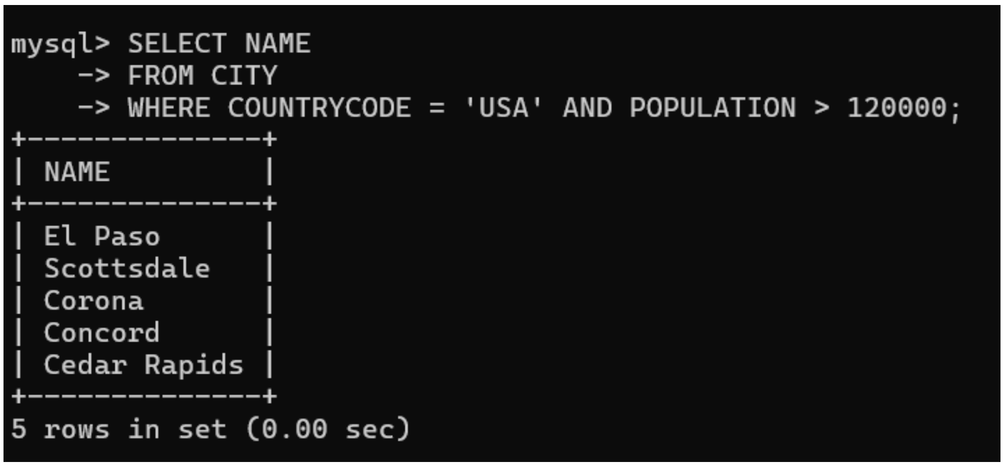
```
mysql> SELECT *
    -> FROM CITY
    -> WHERE COUNTRYCODE = 'USA' AND POPULATION > 100000;
+------+---------------+-------------+------------+------------+
| id   | name          | countrycode | district   | population |
+------+---------------+-------------+------------+------------+
| 3815 | El Paso       | USA         | Texas      |     563662 |
| 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 |
+------+---------------+-------------+------------+------------+
6 rows in set (0.00 sec)
```

Q2. Query the NAME eld for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.

SELECT NAME

FROM CITY

WHERE COUNTRYCODE = 'USA' AND POPULATION > 120000;

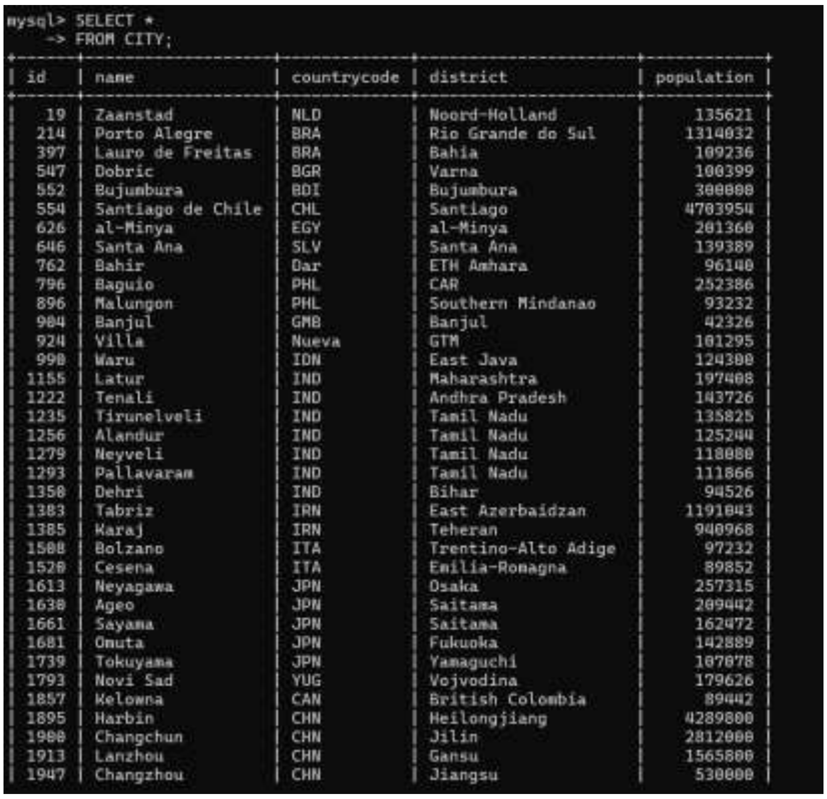


```
mysql> SELECT NAME
    -> FROM CITY
    -> WHERE COUNTRYCODE = 'USA' AND POPULATION > 120000;
+--------------+
| NAME         |
+--------------+
| El Paso      |
| Scottsdale   |
| Corona       |
| Concord      |
| Cedar Rapids |
+--------------+
5 rows in set (0.00 sec)
```

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

SELECT *

FROM CITY;



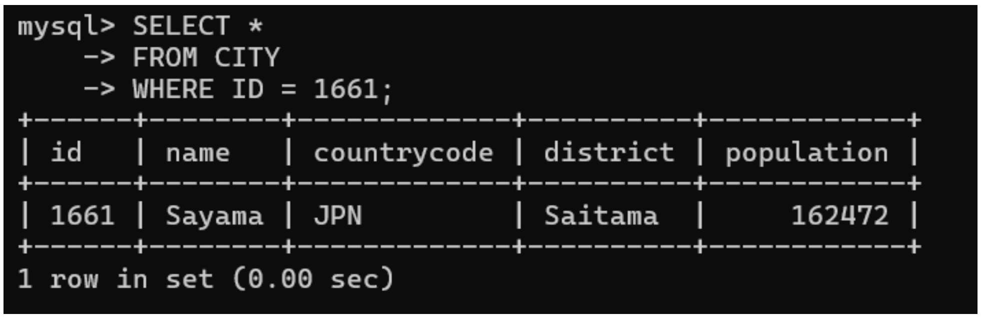
```
mysql> SELECT *
    -> FROM CITY;
+------+-------------------+-------------+----------------------+------------+
| id   | name              | countrycode | district             | population |
+------+-------------------+-------------+----------------------+------------+
|   19 | Zaanstad          | NLD         | Noord-Holland        |     135621 |
|  214 | Porto Alegre      | BRA         | Rio Grande do Sul    |    1314032 |
|  397 | Lauro de Freitas  | BRA         | Bahia                |     109236 |
|  547 | Dobric            | BGR         | Varna                |     100399 |
|  552 | Bujumbura         | BDI         | Bujumbura            |     300000 |
|  554 | Santiago de Chile | CHL         | Santiago             |    4703954 |
|  626 | al-Minya          | EGY         | al-Minya             |     201360 |
|  646 | Santa Ana         | SLV         | Santa Ana            |     139389 |
|  762 | Bahir             | Dar         | ETH Amhara           |      96140 |
|  796 | Baguio            | PHL         | CAR                  |     252386 |
|  896 | Malungon          | PHL         | Southern Mindanao    |      93232 |
|  904 | Banjul            | GMB         | Banjul               |      42326 |
|  924 | Villa             | Nueva       | GTM                  |     101295 |
|  990 | Waru              | IDN         | East Java            |     124300 |
| 1155 | Latur             | IND         | Maharashtra          |     197408 |
| 1222 | Tenali            | IND         | Andhra Pradesh       |     143726 |
| 1235 | Tirunelveli       | IND         | Tamil Nadu           |     135825 |
| 1256 | Alandur           | IND         | Tamil Nadu           |     125244 |
| 1279 | Neyveli           | IND         | Tamil Nadu           |     118080 |
| 1293 | Pallavaram        | IND         | Tamil Nadu           |     111866 |
| 1358 | Dehri             | IND         | Bihar                |      94526 |
| 1383 | Tabriz            | IRN         | East Azerbaidzan     |    1191043 |
| 1385 | Karaj             | IRN         | Teheran              |     940968 |
| 1508 | Bolzano           | ITA         | Trentino-Alto Adige  |      97232 |
| 1520 | Cesena            | ITA         | Emilia-Romagna       |      89852 |
| 1613 | Neyagawa          | JPN         | Osaka                |     257315 |
| 1630 | Ageo              | JPN         | Saitama              |     209442 |
| 1661 | Sayama            | JPN         | Saitama              |     162472 |
| 1681 | Omuta             | JPN         | Fukuoka              |     142889 |
| 1739 | Tokuyama          | JPN         | Yamaguchi            |     107078 |
| 1793 | Novi Sad          | YUG         | Vojvodina            |     179626 |
| 1857 | Kelowna           | CAN         | British Colombia     |      89442 |
| 1895 | Harbin            | CHN         | Heilongjiang         |    4289800 |
| 1900 | Changchun         | CHN         | Jilin                |    2812000 |
| 1913 | Lanzhou           | CHN         | Gansu                |    1565800 |
| 1947 | Changzhou         | CHN         | Jiangsu              |     530000 |
```

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

SELECT *

FROM CITY

WHERE ID = 1661;



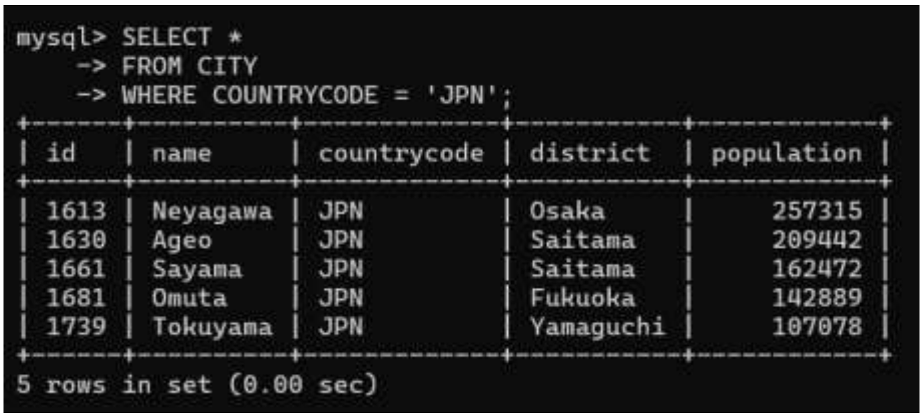
```
mysql> SELECT *
    -> FROM CITY
    -> WHERE ID = 1661;
+------+--------+-------------+----------+------------+
| id   | name   | countrycode | district | population |
+------+--------+-------------+----------+------------+
| 1661 | Sayama | JPN         | Saitama  |     162472 |
+------+--------+-------------+----------+------------+
1 row in set (0.00 sec)
```

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

SELECT *

FROM CITY

WHERE COUNTRYCODE = 'JPN';



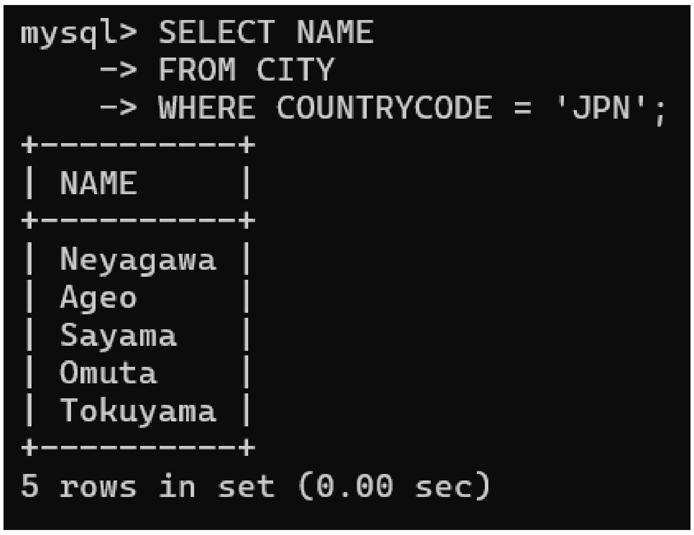
```
mysql> SELECT *
    -> FROM CITY
    -> WHERE COUNTRYCODE = 'JPN';
+------+----------+-------------+-----------+------------+
| id   | name     | countrycode | district  | population |
+------+----------+-------------+-----------+------------+
| 1613 | Neyagawa | JPN         | Osaka     |     257315 |
| 1630 | Ageo     | JPN         | Saitama   |     209442 |
| 1661 | Sayama   | JPN         | Saitama   |     162472 |
| 1681 | Omuta    | JPN         | Fukuoka   |     142889 |
| 1739 | Tokuyama | JPN         | Yamaguchi |     107078 |
+------+----------+-------------+-----------+------------+
5 rows in set (0.00 sec)
```

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

SELECT NAME

FROM CITY

WHERE COUNTRYCODE = 'JPN';

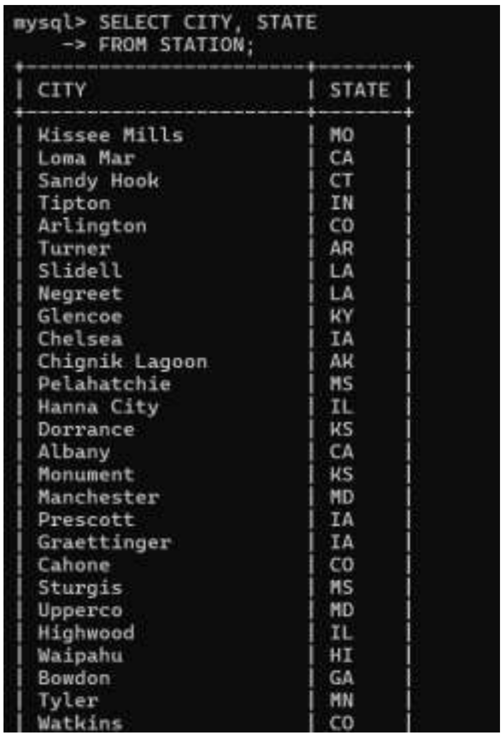


```
mysql> SELECT NAME
    -> FROM CITY
    -> WHERE COUNTRYCODE = 'JPN';
+----------+
| NAME     |
+----------+
| Neyagawa |
| Ageo     |
| Sayama   |
| Omuta    |
| Tokuyama |
+----------+
5 rows in set (0.00 sec)
```

Q7. Query a list of CITY and STATE from the STATION table.

SELECT CITY, STATE

FROM STATION;



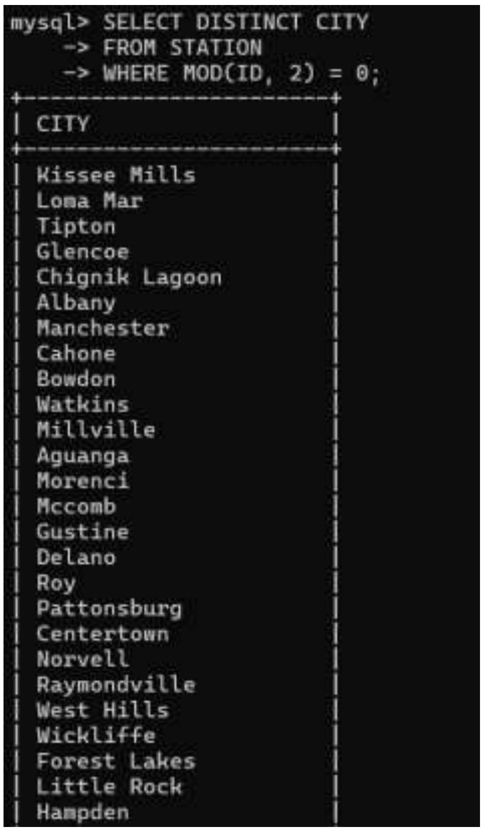
```
mysql> SELECT CITY, STATE
    -> FROM STATION;
+-----------------------+-------+
| CITY                  | STATE |
+-----------------------+-------+
| Kissee Mills          | MO    |
| Loma Mar              | CA    |
| Sandy Hook            | CT    |
| Tipton                | IN    |
| Arlington             | CO    |
| Turner                | AR    |
| Slidell               | LA    |
| Negreet               | LA    |
| Glencoe               | KY    |
| Chelsea               | IA    |
| Chignik Lagoon        | AK    |
| Pelahatchie           | MS    |
| Hanna City            | IL    |
| Dorrance              | KS    |
| Albany                | CA    |
| Monument              | KS    |
| Manchester            | MD    |
| Prescott              | IA    |
| Graettinger           | IA    |
| Cahone                | CO    |
| Sturgis               | MS    |
| Upperco               | MD    |
| Highwood              | IL    |
| Waipahu               | HI    |
| Bowdon                | GA    |
| Tyler                 | MN    |
| Watkins               | CO    |
```

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.

SELECT DISTINCT CITY

FROM STATION

WHERE MOD(ID, 2) = 0;

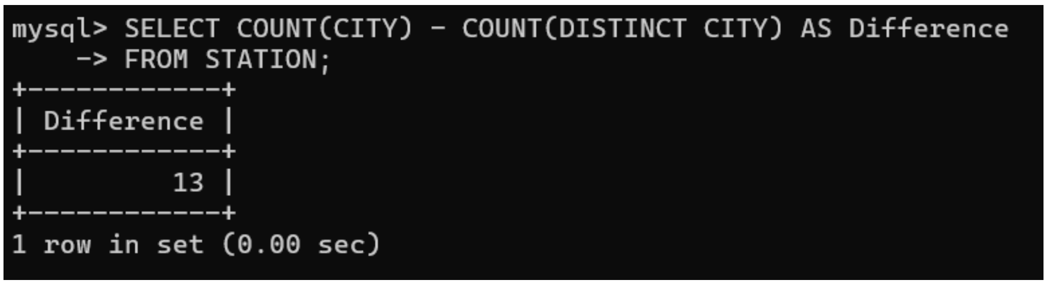


```
mysql> SELECT DISTINCT CITY
    -> FROM STATION
    -> WHERE MOD(ID, 2) = 0;
+-----------------------+
| CITY                  |
+-----------------------+
| Kissee Mills          |
| Loma Mar              |
| Tipton                |
| Glencoe               |
| Chignik Lagoon        |
| Albany                |
| Manchester            |
| Cahone                |
| Bowdon                |
| Watkins               |
| Millville             |
| Aguanga               |
| Morenci               |
| Mccomb                |
| Gustine               |
| Delano                |
| Roy                   |
| Pattonsburg           |
| Centertown            |
| Norvell               |
| Raymondville          |
| West Hills            |
| Wickliffe             |
| Forest Lakes          |
| Little Rock           |
| Hampden               |
```

Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table. For example, if there are three records in the table with CITY values 'New York', 'New York', 'Bengalaru', there are 2 different city names: 'New York' and 'Bengalaru'. The query returns, because total number of records - number of unique city names = 3-2 =1.

SELECT COUNT(CITY) - COUNT(DISTINCT CITY) AS Difference

FROM STATION;



```
mysql> SELECT COUNT(CITY) - COUNT(DISTINCT CITY) AS Difference
    -> FROM STATION;
+------------+
| Difference |
+------------+
|         13 |
+------------+
1 row in set (0.00 sec)
```

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. Sample Input For example, CITY has four entries: DEF, ABC, PQRS and WXY. Sample Output ABC 3 PQRS 4

Hint - When ordered alphabetically, the CITY names are listed as ABC, DEF, PQRS, and WXY, with lengths and. The longest name is PQRS, but there are options for shortest named city. Choose ABC, because it comes rst alphabetically. Note You can write two separate queries to get the desired output. It need not be a single query.

Query for the shortest CITY name:

SELECT CITY, LENGTH(CITY) AS Length

FROM STATION

ORDER BY LENGTH(CITY), CITY

LIMIT 1;

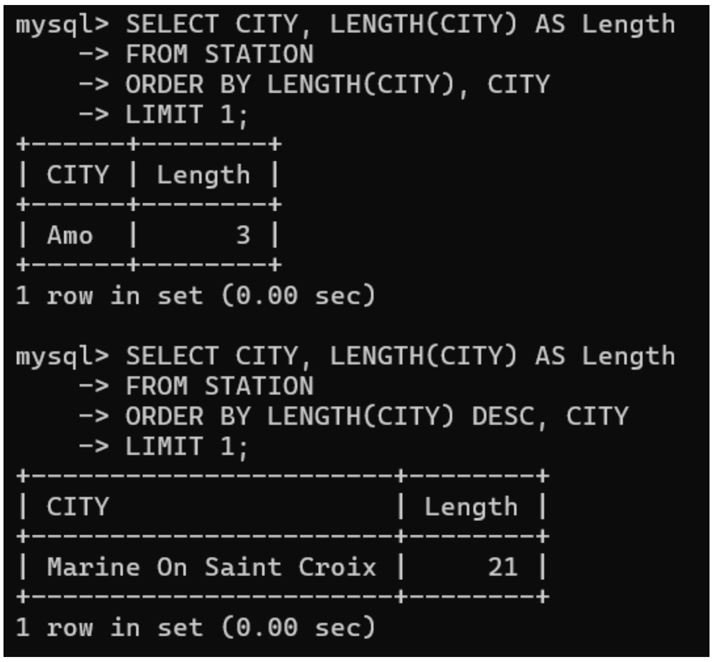
Query for the longest CITY name:

SELECT CITY, LENGTH(CITY) AS Length

FROM STATION

ORDER BY LENGTH(CITY) DESC, CITY

LIMIT 1;



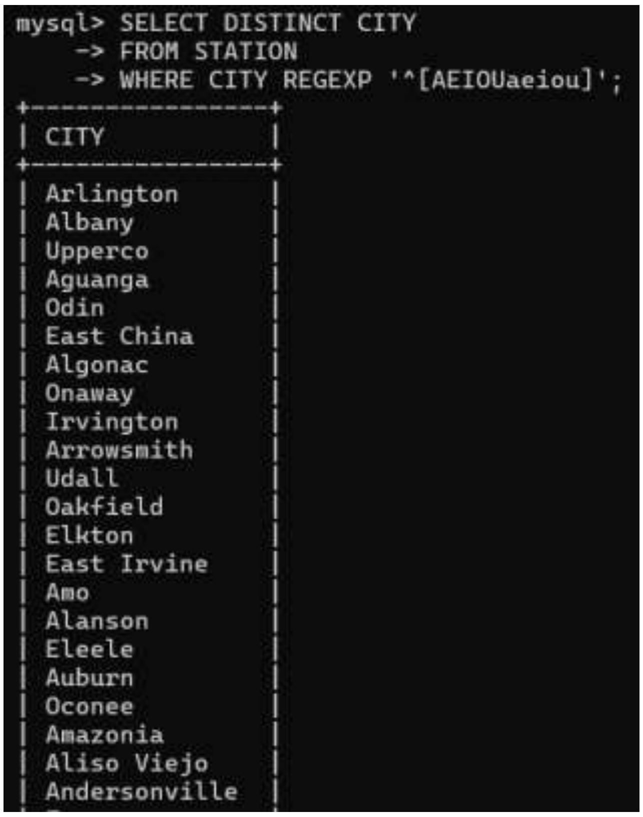
```
mysql> SELECT CITY, LENGTH(CITY) AS Length
    -> FROM STATION
    -> ORDER BY LENGTH(CITY), CITY
    -> LIMIT 1;
+------+--------+
| CITY | Length |
+------+--------+
| Amo  |      3 |
+------+--------+
1 row in set (0.00 sec)

mysql> SELECT CITY, LENGTH(CITY) AS Length
    -> FROM STATION
    -> ORDER BY LENGTH(CITY) DESC, CITY
    -> LIMIT 1;
+-----------------------+--------+
| CITY                  | Length |
+-----------------------+--------+
| Marine On Saint Croix |     21 |
+-----------------------+--------+
1 row in set (0.00 sec)
```

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

SELECT DISTINCT CITY

FROM STATION

WHERE CITY REGEXP '^[AEIOUaeiou]';



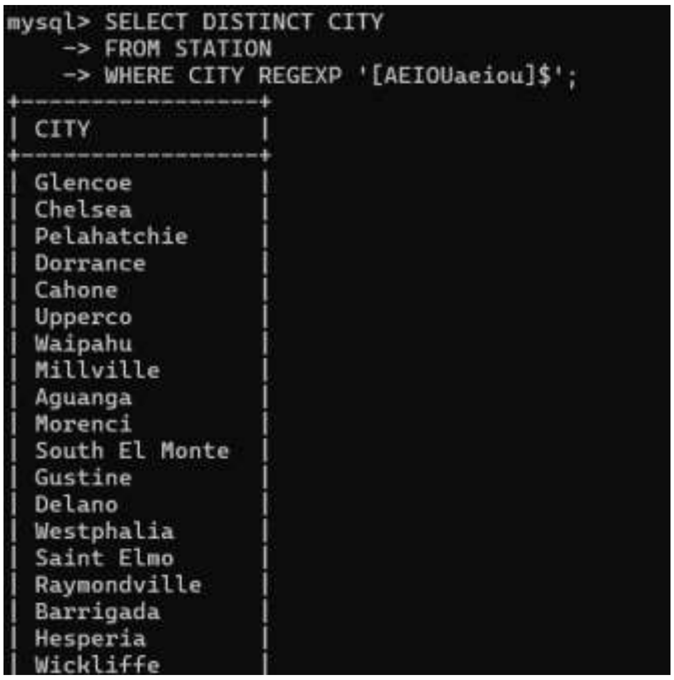
```
mysql> SELECT DISTINCT CITY
    -> FROM STATION
    -> WHERE CITY REGEXP '^[AEIOUaeiou]';
+----------------+
| CITY           |
+----------------+
| Arlington      |
| Albany         |
| Upperco        |
| Aguanga        |
| Odin           |
| East China     |
| Algonac        |
| Onaway         |
| Irvington      |
| Arrowsmith     |
| Udall          |
| Oakfield       |
| Elkton         |
| East Irvine    |
| Amo            |
| Alanson        |
| Eleele         |
| Auburn         |
| Oconee         |
| Amazonia       |
| Aliso Viejo    |
| Andersonville  |
```

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

SELECT DISTINCT CITY

FROM STATION

WHERE CITY REGEXP '[AEIOUaeiou]$';



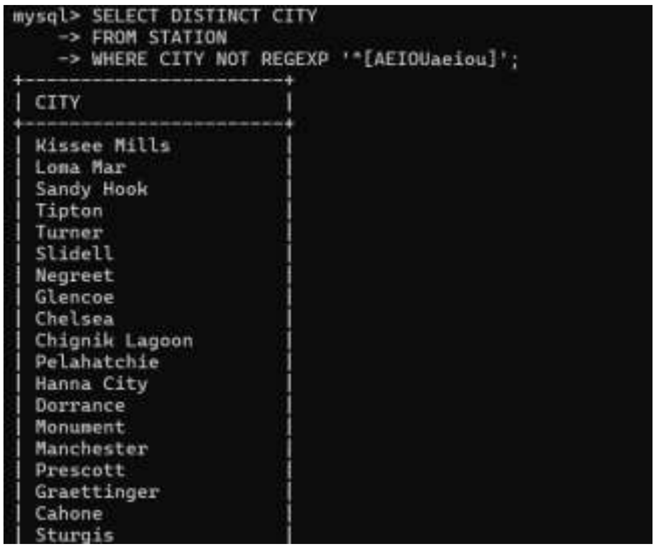
```
mysql> SELECT DISTINCT CITY
    -> FROM STATION
    -> WHERE CITY REGEXP '[AEIOUaeiou]$';
+------------------+
| CITY             |
+------------------+
| Glencoe          |
| Chelsea          |
| Pelahatchie      |
| Dorrance         |
| Cahone           |
| Upperco          |
| Waipahu          |
| Millville        |
| Aguanga          |
| Morenci          |
| South El Monte   |
| Gustine          |
| Delano           |
| Westphalia       |
| Saint Elmo       |
| Raymondville     |
| Barrigada        |
| Hesperia         |
| Wickliffe        |
```

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 REGEXP '^[AEIOUaeiou]';



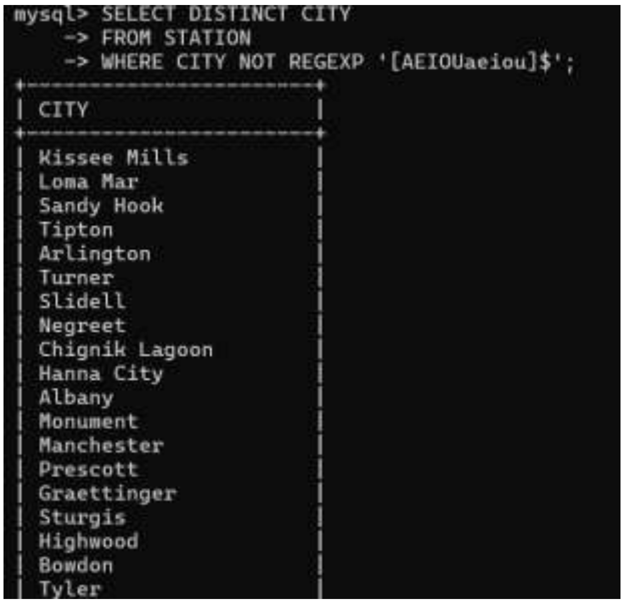
```
mysql> SELECT DISTINCT CITY
    -> FROM STATION
    -> WHERE CITY NOT REGEXP '^[AEIOUaeiou]';
+----------------------+
| CITY                 |
+----------------------+
| Kissee Mills         |
| Loma Mar             |
| Sandy Hook           |
| Tipton               |
| Turner               |
| Slidell              |
| Negreet              |
| Glencoe              |
| Chelsea              |
| Chignik Lagoon       |
| Pelahatchie          |
| Hanna City           |
| Dorrance             |
| Monument             |
| Manchester           |
| Prescott             |
| Graettinger          |
| Cahone               |
| Sturgis              |
```

Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates

SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT REGEXP '[AEIOUaeiou]$';



```
mysql> SELECT DISTINCT CITY
    -> FROM STATION
    -> WHERE CITY NOT REGEXP '[AEIOUaeiou]$';
+------------------------+
| CITY                   |
+------------------------+
| Kissee Mills           |
| Loma Mar               |
| Sandy Hook             |
| Tipton                 |
| Arlington              |
| Turner                 |
| Slidell                |
| Negreet                |
| Chignik Lagoon         |
| Hanna City             |
| Albany                 |
| Monument               |
| Manchester             |
| Prescott               |
| Graettinger            |
| Sturgis                |
| Highwood               |
| Bowdon                 |
| Tyler                  |
```

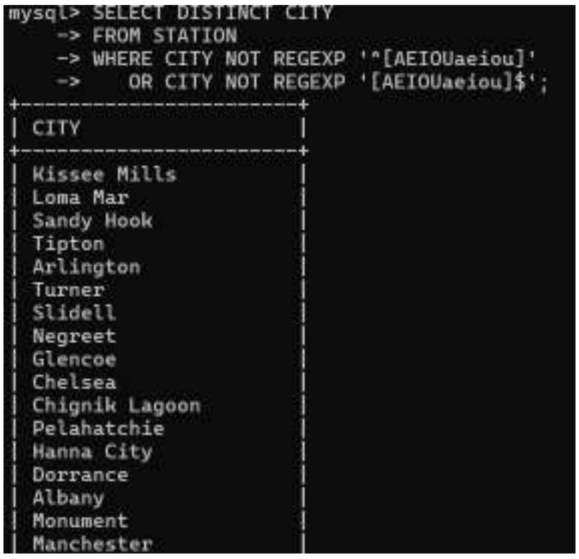
Q15. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT REGEXP '^[AEIOUaeiou]'

   OR CITY NOT REGEXP '[AEIOUaeiou]$';



```
mysql> SELECT DISTINCT CITY
    -> FROM STATION
    -> WHERE CITY NOT REGEXP '^[AEIOUaeiou]'
    ->    OR CITY NOT REGEXP '[AEIOUaeiou]$';
+------------------------+
| CITY                   |
+------------------------+
| Kissee Mills           |
| Loma Mar               |
| Sandy Hook             |
| Tipton                 |
| Arlington              |
| Turner                 |
| Slidell                |
| Negreet                |
| Glencoe                |
| Chelsea                |
| Chignik Lagoon         |
| Pelahatchie            |
| Hanna City             |
| Dorrance               |
| Albany                 |
| Monument               |
| Manchester             |
```

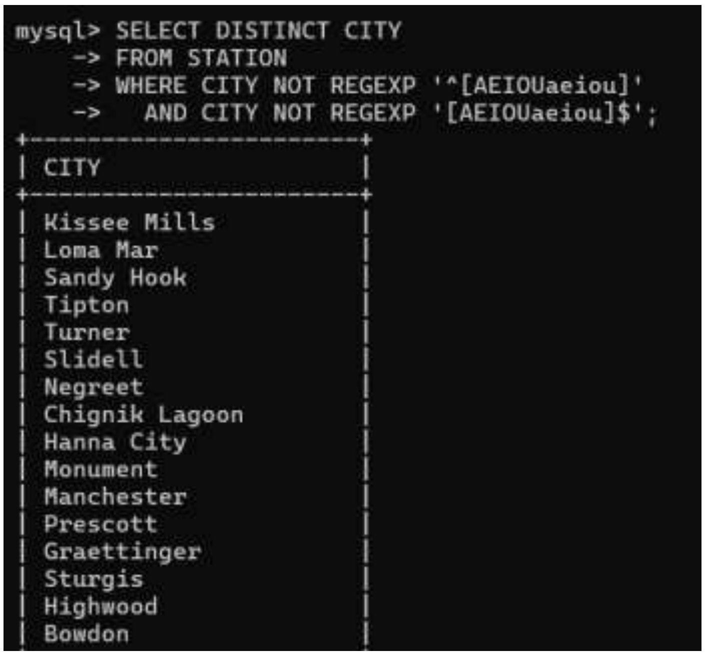
Q16. Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

SELECT DISTINCT CITY

FROM STATION

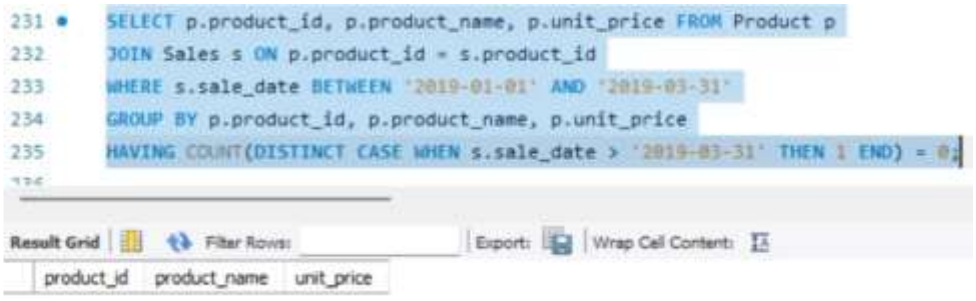
WHERE CITY NOT REGEXP '^[AEIOUaeiou]'

  AND CITY NOT REGEXP '[AEIOUaeiou]$';



```
mysql> SELECT DISTINCT CITY
    -> FROM STATION
    -> WHERE CITY NOT REGEXP '^[AEIOUaeiou]'
    ->   AND CITY NOT REGEXP '[AEIOUaeiou]$';
+-----------------------+
| CITY                  |
+-----------------------+
| Kissee Mills          |
| Loma Mar              |
| Sandy Hook            |
| Tipton                |
| Turner                |
| Slidell               |
| Negreet               |
| Chignik Lagoon        |
| Hanna City            |
| Monument              |
| Manchester            |
| Prescott              |
| Graettinger           |
| Sturgis               |
| Highwood              |
| Bowdon                |
```

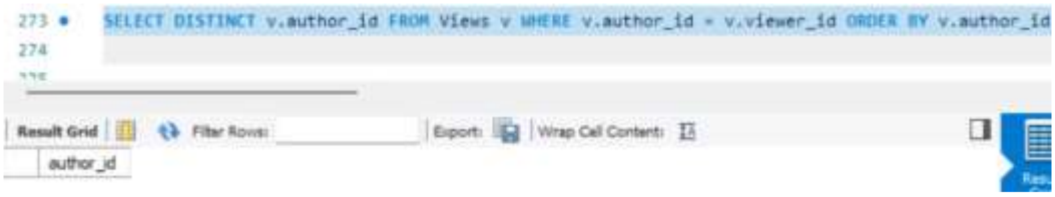
Q17. Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.



```
SELECT p.product_id, p.product_name, p.unit_price FROM Product p
JOIN Sales s ON p.product_id = s.product_id
WHERE s.sale_date BETWEEN '2019-01-01' AND '2019-03-31'
GROUP BY p.product_id, p.product_name, p.unit_price
HAVING COUNT(DISTINCT CASE WHEN s.sale_date > '2019-03-31' THEN 1 END) = 0;
```

product_id	product_name	unit_price

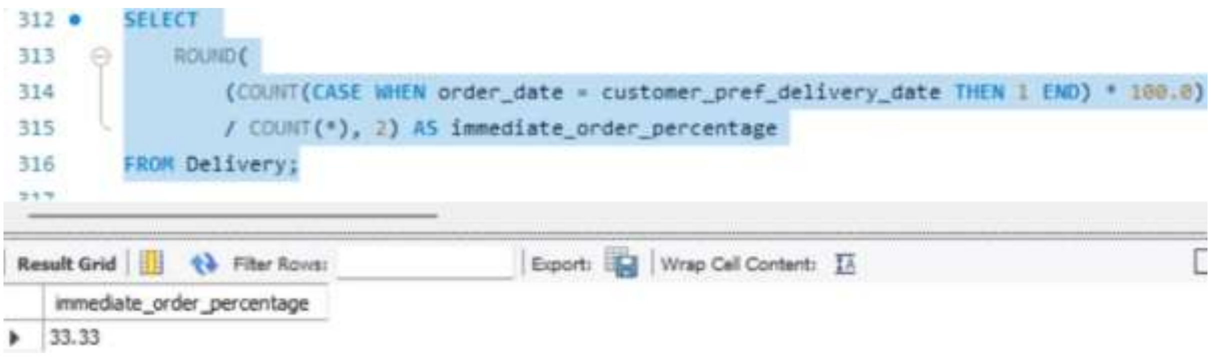
Q18. Write an SQL query to find all the authors that viewed at least one of their own articles. Return the result table sorted by id in ascending order.



```
SELECT DISTINCT v.author_id FROM Views v WHERE v.author_id = v.viewer_id ORDER BY v.author_id
```

author_id

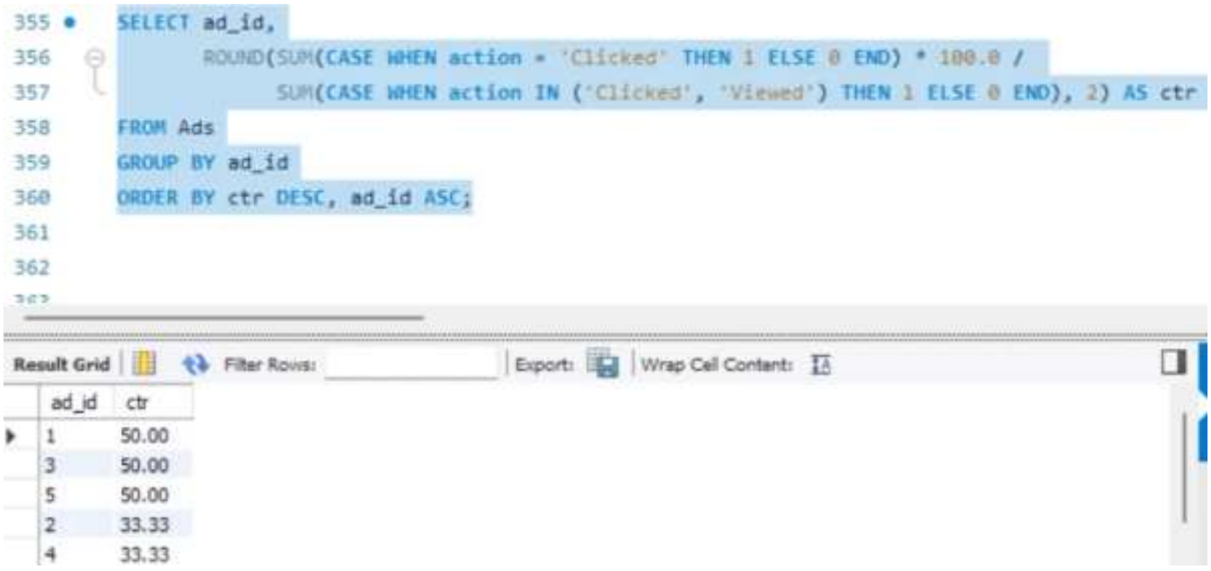
Q19. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.



```sql
SELECT
    ROUND(
        (COUNT(CASE WHEN order_date = customer_pref_delivery_date THEN 1 END) * 100.0)
        / COUNT(*), 2) AS immediate_order_percentage
FROM Delivery;
```

```
immediate_order_percentage
33.33
```

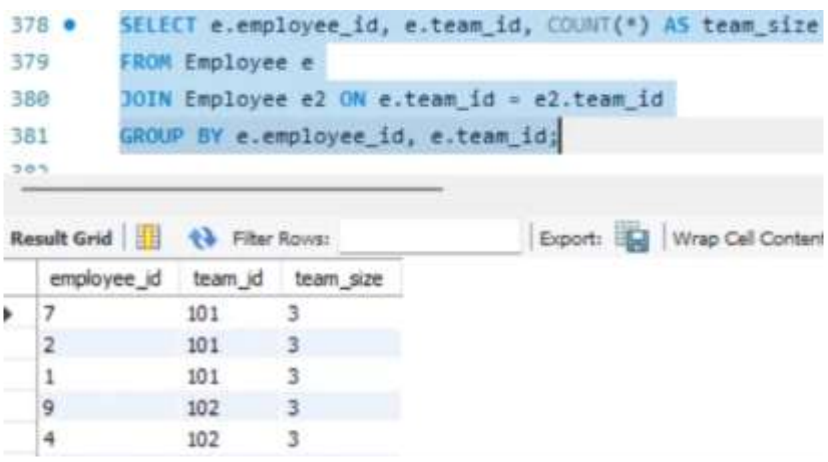
Q20. Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points. Return the result table ordered by ctr in descending order and by ad_id in ascending order in case of a tie.



```sql
SELECT ad_id,
    ROUND(SUM(CASE WHEN action = 'Clicked' THEN 1 ELSE 0 END) * 100.0 /
        SUM(CASE WHEN action IN ('Clicked', 'Viewed') THEN 1 ELSE 0 END), 2) AS ctr
FROM Ads
GROUP BY ad_id
ORDER BY ctr DESC, ad_id ASC;
```

```
ad_id  ctr
1      50.00
3      50.00
5      50.00
2      33.33
4      33.33
```

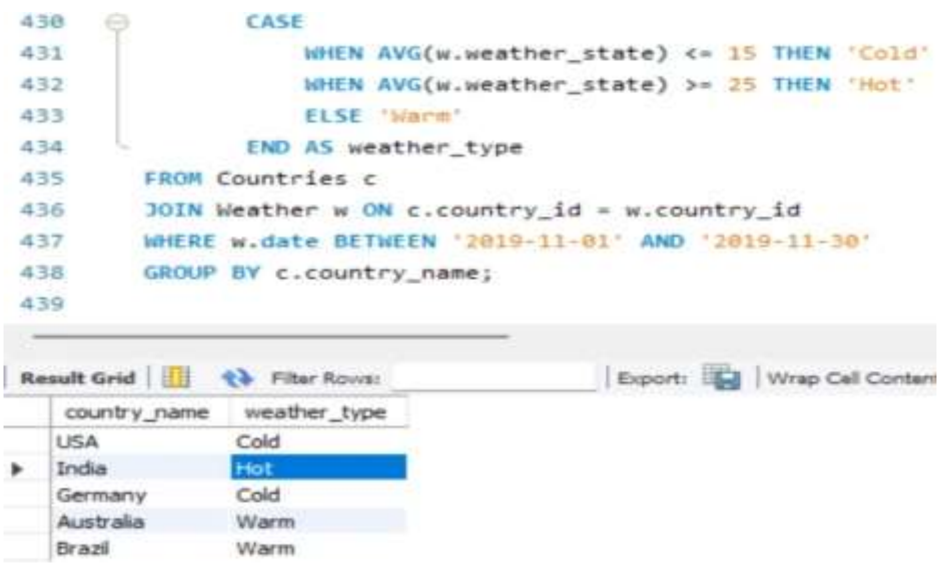
Q21. Write an SQL query to find the team size of each of the employees.



```sql
SELECT e.employee_id, e.team_id, COUNT(*) AS team_size
FROM Employee e
JOIN Employee e2 ON e.team_id = e2.team_id
GROUP BY e.employee_id, e.team_id;
```

```
employee_id  team_id  team_size
7            101      3
2            101      3
1            101      3
9            102      3
4            102      3
```

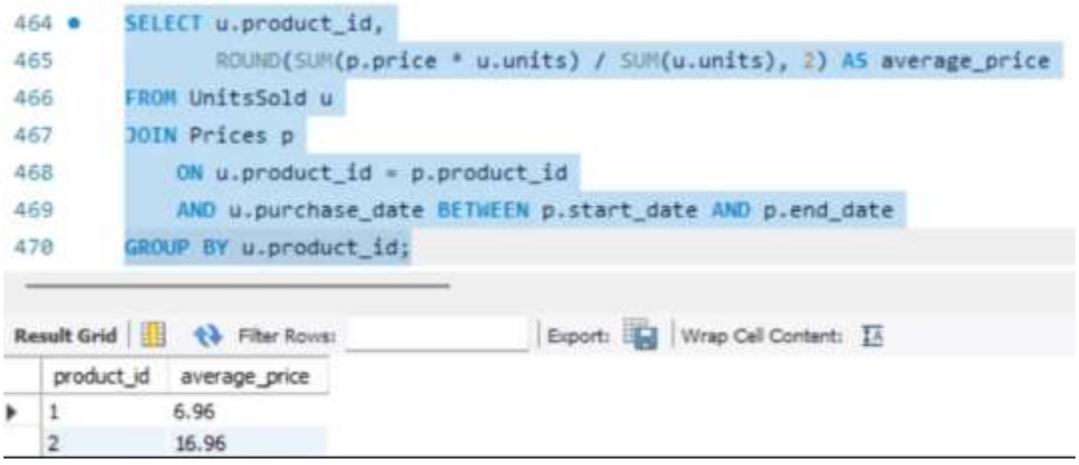
Q22. Write an SQL query to find the type of weather in each country for November 2019.



```
        CASE
            WHEN AVG(w.weather_state) <= 15 THEN 'Cold'
            WHEN AVG(w.weather_state) >= 25 THEN 'Hot'
            ELSE 'Warm'
        END AS weather_type
    FROM Countries c
    JOIN Weather w ON c.country_id = w.country_id
    WHERE w.date BETWEEN '2019-11-01' AND '2019-11-30'
    GROUP BY c.country_name;
```

country_name	weather_type
USA	Cold
India	Hot
Germany	Cold
Australia	Warm
Brazil	Warm

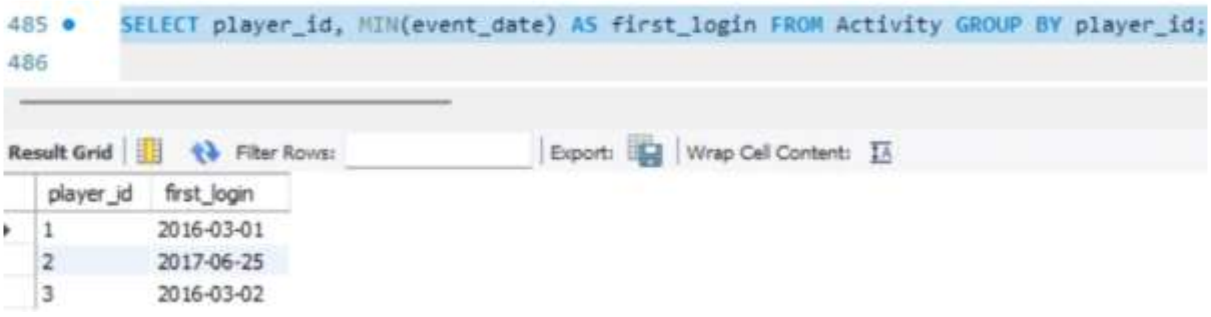
Q23. Write an SQL query to find the average selling price for each product. average_price should be rounded to 2 decimal places.



```
SELECT u.product_id,
       ROUND(SUM(p.price * u.units) / SUM(u.units), 2) AS average_price
FROM UnitsSold u
JOIN Prices p
  ON u.product_id = p.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

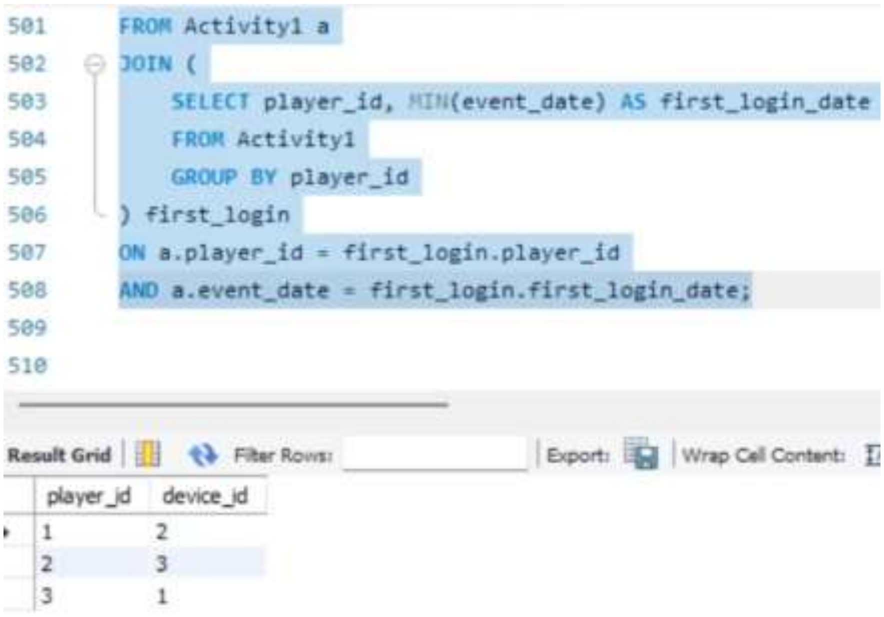
Q24. Write an SQL query to report the first login date for each player.



```
SELECT player_id, MIN(event_date) AS first_login FROM Activity GROUP BY player_id;
```

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

Q25. Write an SQL query to report the device that is first logged in for each player.



```
FROM Activity1 a
JOIN (
    SELECT player_id, MIN(event_date) AS first_login_date
    FROM Activity1
    GROUP BY player_id
) first_login
ON a.player_id = first_login.player_id
AND a.event_date = first_login.first_login_date;
```

player_id	device_id
1	2
2	3
3	1