`SQL for Data Analytics - Practice Problems Solved`

In [1]:
```
import requests
import mysql.connector
# import ast
# import sys
import pandas as pd
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
connection = mysql.connector.connect(host='localhost',
                  port='3306',
                  user='root',
                  password='**',
                  database='sale',
                  auth_plugin='mysql_native_password')
```

In [4]:
```
connection
```

Out[4]:
```
<mysql.connector.connection.MySQLConnection at 0x27dbdef0b80>
```

***Sample Questions***

*You need to write the SQL query between the space mentioned in the box below and then press Ctrl+Enter to run the query. Here is an example*

In [7]:
```
pd.read_sql_query('SHOW TABLES',connection)
```

Out[7]:

| | Tables_in_sale |
|---|---|
| **0** | customer_dim |
| **1** | delivery_person_dim |
| **2** | order_dim |
| **3** | pincode_dim |
| **4** | product_dim |

***Q0**. How many customers are male and how many are female?*

In [8]:
```
pd.read_sql_query('''
SELECT
    c.gender,
    COUNT(c.cust_id) AS cust_cnt
FROM customer_dim AS c
GROUP BY c.gender

''',connection)
```

Out[8]:

| | gender | cust_cnt |
|---|---|---|
| **0** | male | 10 |
| **1** | female | 4 |

***Q1**. How many customers do not have DOB information available?*

In [9]:
```
pd.read_sql_query('''
SELECT
    COUNT(c.cust_id) AS cust_cnt
FROM customer_dim AS c
WHERE c.dob IS NULL

''',connection)
```

Out[9]:

| | cust_cnt |
|---|---|
| **0** | 0 |

***Q2**. How many customers are there in each pincode and gender combination?*

In [10]:
```
pd.read_sql_query('''
SELECT
    c.primary_pincode,
    c.gender,
    COUNT(c.cust_id) AS cust_cnt
FROM customer_dim AS c
GROUP BY
    c.primary_pincode,
    c.gender

''',connection)
```

Out[10]:

|   | primary_pincode | gender | cust_cnt |
|---|---|---|---|
| **0** | 110001 | male | 3 |
| **1** | 400001 | male | 2 |
| **2** | 560001 | male | 1 |
| **3** | 600001 | female | 1 |
| **4** | 500001 | male | 1 |
| **5** | 700001 | male | 2 |
| **6** | 560001 | female | 1 |
| **7** | 600001 | male | 1 |
| **8** | 500001 | female | 1 |
| **9** | 700001 | female | 1 |

***Q3**. Print product name and mrp for products which have more than 50000 MRP*

In [11]:
```
pd.read_sql_query('''
SELECT
    p.product_name,
    p.mrp
FROM product_dim AS p
WHERE p.mrp>50000

''',connection)
```

Out[11]:

|   | product_name | mrp |
|---|---|---|
| **0** | HP 241H | 80000 |
| **1** | Dell AX420 | 75000 |

***Q4**. How many delivery personal are there in each pincode?*

In [12]:
```
pd.read_sql_query('''
SELECT
    d.pincode,
    COUNT(d.delivery_person_id) AS dp_cnt
FROM delivery_person_dim AS d
GROUP BY d.pincode

''',connection)
```

Out[12]:

|   | pincode | dp_cnt |
|---|---|---|
| **0** | 110001 | 1 |
| **1** | 400001 | 4 |
| **2** | 500001 | 1 |
| **3** | 560001 | 1 |
| **4** | 600001 | 1 |
| **5** | 700001 | 2 |

***Q5**. For each Pin code, print the count of orders, sum of total amount paid, average amount paid, maximum amount paid, minimum amount paid for the transactions which were paid by 'cash'. Take only 'buy' order types*

In [13]:
```
pd.read_sql_query('''
SELECT
    o.delivery_pincode,
    COUNT(o.order_id) AS order_cnt,
    SUM(o.total_amount_paid) AS total_amount_paid,
    AVG(o.total_amount_paid) AS avg_amount_paid,
    MAX(o.total_amount_paid) AS max_amount_paid,
    MIN(o.total_amount_paid) AS min_amount_paid
FROM order_dim as o
WHERE o.payment_type = 'cash'
    AND o.order_type = 'buy'
GROUP BY
    o.delivery_pincode

''',connection)
```

Out[13]:

| | delivery_pincode | order_cnt | total_amount_paid | avg_amount_paid | max_amount_paid | min_amount_paid |
|---|---|---|---|---|---|---|
| **0** | 500001 | 28 | 4798422.0 | 171372.2143 | 646800 | 1314 |
| **1** | 700001 | 53 | 6871936.0 | 129659.1698 | 721280 | 687 |
| **2** | 600001 | 19 | 1456296.0 | 76647.1579 | 669600 | 1213 |
| **3** | 400001 | 105 | 11546300.0 | 109964.7619 | 669750 | 644 |
| **4** | 110001 | 19 | 4026734.0 | 211933.3684 | 608103 | 676 |
| **5** | 560001 | 19 | 2829381.0 | 148914.7895 | 609120 | 662 |

***Q6***. _For each delivery_person_id, print the count of orders and total amount paid for product*id = 12350 or 12348 and total units > 8. Sort the output by total amount paid in descending order. Take only 'buy' order types*

In [14]:
```
pd.read_sql_query('''
SELECT
    o.delivery_person_id,
    COUNT(o.order_id) AS order_cnt,
    SUM(o.total_amount_paid) AS total_amount_paid
FROM order_dim as o
WHERE o.product_id IN (12350,12348)
    AND o.tot_units > 8
    AND o.order_type = 'buy'
GROUP BY
    o.delivery_person_id
ORDER BY
    SUM(o.total_amount_paid) DESC

''',connection)
```

Out[14]:

| | delivery_person_id | order_cnt | total_amount_paid |
|---|---|---|---|
| **0** | 1000002 | 10 | 76801.0 |
| **1** | 1000010 | 7 | 56285.0 |
| **2** | 1000003 | 6 | 52828.0 |
| **3** | 1000001 | 6 | 51653.0 |
| **4** | 1000009 | 6 | 49142.0 |
| **5** | 1000008 | 6 | 48424.0 |
| **6** | 1000005 | 5 | 43677.0 |
| **7** | 1000007 | 5 | 41535.0 |
| **8** | 1000004 | 5 | 35452.0 |
| **9** | 1000006 | 4 | 32915.0 |

***Q7**. Print the Full names (first name plus last name) for customers that have email on "gmail.com"?*

In [15]:
```
pd.read_sql_query('''
SELECT
    c.first_name || ' ' || c.last_name AS name
FROM customer_dim AS c
WHERE c.email LIKE '%gmail%'

''',connection)
```

Out[15]:

| | name |
|---|---|
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |
| **5** | 0 |
| **6** | 0 |
| **7** | 0 |

***Q8**. How many orders had #units between 1-3, 4-6 and 7+? Take only 'buy' order types*

In [16]:
```
pd.read_sql_query('''
SELECT
    CASE
        WHEN o.tot_units <= 3 THEN '1. 1-3'
        WHEN o.tot_units <= 6 THEN '2. 4-6'
    ELSE '3. 7+' END AS unit_bkt,
    COUNT(o.order_id) AS order_cnt
FROM order_dim AS o
WHERE o.order_type = 'buy'
GROUP BY
```

```
    CASE
        WHEN o.tot_units <= 3 THEN '1. 1-3'
        WHEN o.tot_units <= 6 THEN '2. 4-6'
    ELSE '3. 7+' END

''',connection)
```

Out[16]:

|   | unit_bkt | order_cnt |
|---|---|---|
| **0** | 1. 1-3 | 314 |
| **1** | 3. 7+ | 372 |
| **2** | 2. 4-6 | 314 |

***Q9**. Which pincode has average amount paid more than 150,000? Take only 'buy' order types*

In [17]:

```
pd.read_sql_query('''
SELECT
    o.delivery_pincode,
    AVG(total_amount_paid) AS avg_amount_paid
FROM order_dim AS o
WHERE o.order_type = 'buy'
GROUP BY o.delivery_pincode
HAVING AVG(total_amount_paid)>150000

''',connection)
```

Out[17]:

|   | delivery_pincode | avg_amount_paid |
|---|---|---|
| **0** | 110001 | 158145.6505 |

***Q10**.* _Create following columns from order*dim data:*

- *order date*
- *Order day*
- *Order month*
- *Order year*

In [18]:

```
pd.read_sql_query('''
SELECT
    o.order_date,
    SUBSTR(o.order_date, 1, 2) AS order_day,
    SUBSTR(o.order_date, 4, 2) AS order_month,
    SUBSTR(o.order_date, 7, 4) AS order_year
FROM order_dim AS o
WHERE o.order_type = 'buy'

''',connection)
```

Out[18]:

| | order_date | order_day | order_month | order_year |
|---|---|---|---|---|
| **0** | 01-01-2020 | 01 | 01 | 2020 |
| **1** | 01-01-2020 | 01 | 01 | 2020 |
| **2** | 01-01-2020 | 01 | 01 | 2020 |
| **3** | 01-01-2020 | 01 | 01 | 2020 |
| **4** | 01-01-2020 | 01 | 01 | 2020 |
| **...** | ... | ... | ... | ... |
| **995** | 01-10-2020 | 01 | 10 | 2020 |
| **996** | 01-10-2020 | 01 | 10 | 2020 |
| **997** | 01-10-2020 | 01 | 10 | 2020 |
| **998** | 01-10-2020 | 01 | 10 | 2020 |
| **999** | 01-10-2020 | 01 | 10 | 2020 |

1000 rows × 4 columns

***Q11**. How many total orders were there in each month and how many of them were returned? Add a column for return rate too. return rate = (100.0 * total return orders) / total buy orders Hint: You will need to combine SUM() with CASE WHEN*

In [19]:
```
pd.read_sql_query('''
SELECT
    SUBSTR(o.order_date, 4, 2) AS order_month,
    SUM(CASE WHEN o.order_type = 'buy' THEN 1 ELSE 0 END) AS tot_buy_orders,
    SUM(CASE WHEN o.order_type = 'return' THEN 1 ELSE 0 END) AS tot_return_orde
    100.0*SUM(CASE WHEN o.order_type = 'return' THEN 1 ELSE 0 END)/SUM(CASE WHI
FROM order_dim AS o
GROUP BY SUBSTR(o.order_date, 4, 2)

''',connection)
```

Out[19]:

| | order_month | tot_buy_orders | tot_return_orders | return_rate |
|---|---|---|---|---|
| **0** | 01 | 119.0 | 3.0 | 2.52101 |
| **1** | 02 | 107.0 | 7.0 | 6.54206 |
| **2** | 03 | 103.0 | 6.0 | 5.82524 |
| **3** | 04 | 115.0 | 6.0 | 5.21739 |
| **4** | 05 | 117.0 | 8.0 | 6.83761 |
| **5** | 06 | 106.0 | 3.0 | 2.83019 |
| **6** | 07 | 110.0 | 4.0 | 3.63636 |
| **7** | 08 | 109.0 | 5.0 | 4.58716 |
| **8** | 09 | 109.0 | 5.0 | 4.58716 |
| **9** | 10 | 5.0 | 3.0 | 60.00000 |

***Q12**. How many units have been sold by each brand? Also get total returned units for each brand.*

In [20]:
```
pd.read_sql_query('''
SELECT
    p.brand,
    SUM(CASE WHEN o.order_type = 'buy' THEN o.tot_units ELSE 0 END) AS total_s
    SUM(CASE WHEN o.order_type = 'return' THEN o.tot_units ELSE 0 END) AS tota
FROM product_dim AS p
LEFT JOIN order_dim AS o
    ON p.product_id = o.product_id
GROUP BY p.brand

''',connection)
```

Out[20]:

| | brand | total_sold | total_returned |
|---|---|---|---|
| **0** | HP | 2666.0 | 145.0 |
| **1** | Dell | 2701.0 | 112.0 |

***Q13***. *How many distinct customers and delivery boys are there in each state?*

In [21]:
```
pd.read_sql_query('''
SELECT
    pin.state,
    COUNT(DISTINCT c.cust_id) AS cust_cnt,
    COUNT(DISTINCT d.delivery_person_id) AS dc_cnt
FROM pincode_dim AS pin

LEFT JOIN customer_dim AS c
    ON pin.pincode = c.primary_pincode

LEFT JOIN delivery_person_dim AS d
    ON pin.pincode = d.pincode

GROUP BY pin.state

''',connection)
```

Out[21]:

| | state | cust_cnt | dc_cnt |
|---|---|---|---|
| **0** | Karnataka | 2 | 1 |
| **1** | Maharastra | 2 | 4 |
| **2** | New Delhi | 3 | 1 |
| **3** | Tamil Nadu | 2 | 1 |
| **4** | Telangana | 2 | 1 |
| **5** | West Bengal | 3 | 2 |

***Q14***. _For every customer, print how many total units were ordered, how many units were ordered from their primary_pincode and how many were ordered not from the primary_pincode. Also calulate the percentage of total units which were ordered from primary*pincode(remember to multiply the numerator by 100.0). Sort by the percentage column in descending order.*

In [22]:
```
pd.read_sql_query('''
SELECT
```

```
    c.cust_id,
    SUM(o.tot_units) AS tot_units,
    SUM(CASE WHEN c.primary_pincode = o.delivery_pincode THEN o.tot_units ELSE
    SUM(CASE WHEN c.primary_pincode != o.delivery_pincode THEN o.tot_units ELSI
    100.0 * SUM(CASE WHEN c.primary_pincode = o.delivery_pincode THEN o.tot_un
FROM customer_dim AS c
LEFT JOIN order_dim AS o
    ON c.cust_id = o.cust_id
GROUP BY c.cust_id
ORDER BY perc_same_city DESC

''',connection)
```

Out[22]:

|  | cust_id | tot_units | units_same_city | units_diff_city | perc_same_city |
|---|---|---|---|---|---|
| 0 | 10000002 | 372.0 | 164.0 | 208.0 | 44.08602 |
| 1 | 10000008 | 410.0 | 152.0 | 258.0 | 37.07317 |
| 2 | 10000012 | 534.0 | 109.0 | 425.0 | 20.41199 |
| 3 | 10000007 | 369.0 | 72.0 | 297.0 | 19.51220 |
| 4 | 10000005 | 375.0 | 59.0 | 316.0 | 15.73333 |
| 5 | 10000006 | 290.0 | 44.0 | 246.0 | 15.17241 |
| 6 | 10000003 | 413.0 | 61.0 | 352.0 | 14.76998 |
| 7 | 10000009 | 537.0 | 66.0 | 471.0 | 12.29050 |
| 8 | 10000004 | 398.0 | 48.0 | 350.0 | 12.06030 |
| 9 | 10000014 | 353.0 | 42.0 | 311.0 | 11.89802 |
| 10 | 10000011 | 356.0 | 35.0 | 321.0 | 9.83146 |
| 11 | 10000013 | 331.0 | 28.0 | 303.0 | 8.45921 |
| 12 | 10000010 | 395.0 | 31.0 | 364.0 | 7.84810 |
| 13 | 10000001 | 491.0 | 29.0 | 462.0 | 5.90631 |

***Q15**. For each product name, print the sum of number of units, total amount paid, total displayed selling price, total mrp of these units, and finally the net discount from selling price (i.e. 100.0 - 100.0 total amount paid / total displayed selling price) AND the net discount from mrp (i.e. 100.0 - 100.0 total amount paid / total mrp)*

In [23]:

```
pd.read_sql_query('''
SELECT
    p.product_name,
    SUM(o.tot_units) AS tot_units,
    SUM(o.total_amount_paid) AS total_amount_paid,
    SUM(o.displayed_selling_price_per_unit * o.tot_units) AS total_displayed_se
    SUM(p.mrp * o.tot_units) AS total_mrp,
    100.0 - 100.0 * SUM(o.total_amount_paid)/SUM(o.displayed_selling_price_per_
    100.0 - 100.0 * SUM(o.total_amount_paid)/SUM(p.mrp * o.tot_units) AS discou
FROM product_dim AS p
LEFT JOIN order_dim AS o
    ON p.product_id = o.product_id
GROUP BY p.product_name

''',connection)
```

Out[23]:

| | product_name | tot_units | total_amount_paid | total_displayed_selling_price | total_mrp | discount_from_s |
|---|---|---|---|---|---|---|
| **0** | HP XYZ Mouse | 1023.0 | 1155504.0 | 1372470.0 | 1534500.0 | 15.8084 |
| **1** | HP 241H | 884.0 | 51396664.0 | 63275200.0 | 70720000.0 | 18.7728 |
| **2** | HP 8GB Pendrive | 904.0 | 578605.0 | 654288.0 | 723200.0 | 11.5672 |
| **3** | Dell ABC Mouse | 942.0 | 809662.0 | 928510.0 | 1036200.0 | 12.7998 |
| **4** | Dell AX420 | 982.0 | 58124196.0 | 65829000.0 | 73650000.0 | 11.7042 |
| **5** | Dell 8GB Pendrive | 889.0 | 574506.0 | 670592.0 | 755650.0 | 14.3285 |

***Q16***. _For every order*id (exclude returns), get the product name and calculate the discount percentage from selling price. Sort by highest discount and print only those rows where discount percentage was above 10.10%.*

In [24]:
```
pd.read_sql_query('''
SELECT
    o.order_id,
    p.product_name,
    100.0 - 100.0 * o.total_amount_paid/(o.displayed_selling_price_per_unit * (
FROM product_dim AS p
LEFT JOIN order_dim AS o
    ON p.product_id = o.product_id
WHERE o.order_type = 'buy'
GROUP BY
    o.order_id,
    p.product_name
HAVING discount_from_sp>10.10

''',connection)
```

Out[24]:

| | order_id | product_name | discount_from_sp |
|---|---|---|---|
| **0** | 10000000133 | Dell 8GB Pendrive | 10.12312 |
| **1** | 10000000150 | Dell 8GB Pendrive | 10.10702 |
| **2** | 10000000155 | Dell 8GB Pendrive | 10.12312 |
| **3** | 10000000160 | HP 8GB Pendrive | 10.11905 |
| **4** | 10000000409 | HP 8GB Pendrive | 10.10638 |
| **5** | 10000000575 | HP 8GB Pendrive | 10.10101 |
| **6** | 10000000580 | Dell 8GB Pendrive | 10.11080 |
| **7** | 10000000653 | HP 8GB Pendrive | 10.11905 |
| **8** | 10000000902 | Dell ABC Mouse | 10.10101 |
| **9** | 10000000997 | Dell 8GB Pendrive | 10.10230 |

***Q17***. _Using the per unit procurement cost in product*dim, find which product category has made the most profit in both absolute amount and percentage Absolute Profit = Total Amt Sold - Total*

*Procurement Cost Percentage Profit = 100.0 * Total Amt Sold / Total Procurement Cost - 100.0*

In [25]:
```
pd.read_sql_query('''
SELECT
    p.category,
    SUM(o.total_amount_paid) - SUM(p.procurement_cost_per_unit * o.tot_units) /
    100.0 * SUM(o.total_amount_paid)/SUM(p.procurement_cost_per_unit * o.tot_un
FROM product_dim AS p
LEFT JOIN order_dim AS o
    ON p.product_id = o.product_id
GROUP BY
    p.category

''',connection)
```

Out[25]:

| | category | abs_profit | perc_profit |
|---|---|---|---|
| **0** | mouse | 970516.0 | 196.57362 |
| **1** | laptop | 40280860.0 | 157.17571 |
| **2** | pendrive | 614461.0 | 213.07426 |

***Q18***. _For every delivery person(use their name), print the total number of order ids (exclude returns) by month in seperate columns i.e. there should be one row for each delivery_person*id and 12 columns for every month in the year*

In [ ]:
```
pd.read_sql_query('''
SELECT
    d.name,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '01' THEN 1 ELSE 0 END) AS Jan,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '02' THEN 1 ELSE 0 END) AS Feb,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '03' THEN 1 ELSE 0 END) AS Mar,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '04' THEN 1 ELSE 0 END) AS Apr,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '05' THEN 1 ELSE 0 END) AS May,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '06' THEN 1 ELSE 0 END) AS Jun,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '07' THEN 1 ELSE 0 END) AS Jul,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '08' THEN 1 ELSE 0 END) AS Aug,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '09' THEN 1 ELSE 0 END) AS Sep,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '10' THEN 1 ELSE 0 END) AS Oct,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '11' THEN 1 ELSE 0 END) AS Nov,
    SUM(CASE WHEN SUBSTR(o.order_date, 4, 2) = '12' THEN 1 ELSE 0 END) AS Dec
FROM delivery_person_dim AS d
INNER JOIN order_dim AS o
    ON d.delivery_person_id = o.delivery_person_id
WHERE o.order_type = 'buy'
GROUP BY
    d.name

''',connection)
```

***Q19***. *For each gender - male and female - find the absolute and percentage profit (like in Q16) by product name.*

In [31]:
```
pd.read_sql_query('''
SELECT
    c.gender,
    p.category,
    SUM(o.total_amount_paid) - SUM(p.procurement_cost_per_unit * o.tot_units) /
    100.0 * SUM(o.total_amount_paid)/SUM(p.procurement_cost_per_unit * o.tot_un
FROM customer_dim AS c

LEFT JOIN order_dim AS o
    ON c.cust_id = o.cust_id

LEFT JOIN product_dim AS p
    ON p.product_id = o.product_id

GROUP BY
    c.gender,
    p.category

''',connection)
```

Out[31]:

| | gender | category | abs_profit | perc_profit |
|---|---|---|---|---|
| **0** | male | laptop | 27828246.0 | 158.55751 |
| **1** | male | mouse | 763260.0 | 203.31324 |
| **2** | male | pendrive | 449435.0 | 216.13187 |
| **3** | female | laptop | 12452614.0 | 154.30808 |
| **4** | female | mouse | 207256.0 | 177.81955 |
| **5** | female | pendrive | 165026.0 | 205.50274 |

***Q20***. *Generally the more numbers of units you buy, the more discount seller will give you. For 'Dell AX420' is there a relationship between number of units ordered and average discount from selling price? Take only 'buy' order types*

In [32]:

```
pd.read_sql_query('''
SELECT
    o.tot_units,
    COUNT(order_id) AS total_orders,
    100.0 - 100.0 * o.total_amount_paid/(o.displayed_selling_price_per_unit * o
FROM product_dim AS p
LEFT JOIN order_dim AS o
    ON p.product_id = o.product_id
WHERE o.order_type = 'buy'
    AND p.product_name = 'Dell AX420'
GROUP BY
    o.tot_units

''',connection)
```

Out[32]:

| | tot_units | total_orders | discount_from_sp |
|---|---|---|---|
| **0** | 2 | 16 | 6.0 |
| **1** | 9 | 19 | 5.0 |
| **2** | 1 | 21 | 6.0 |
| **3** | 3 | 19 | 5.0 |
| **4** | 5 | 19 | 1.0 |
| **5** | 4 | 16 | 5.0 |
| **6** | 6 | 16 | 8.0 |
| **7** | 7 | 18 | 1.0 |
| **8** | 10 | 16 | 8.0 |
| **9** | 8 | 16 | 10.0 |