Assignment 1:SQL
Student: Aman Kaushik:DSI

--SELECT
--Write a query that returns everything in the customer table.

```
SELECT *
FROM customer;
```

--Write a query that displays all of the columns and 10 rows from the customer table,
---sorted by customer_last_name, then customer_first_ name.

```
SELECT *
FROM customer
ORDER BY customer_last_name, customer_first_name
LIMIT 10;
```

---WHERE
--Write a query that returns all customer purchases of product IDs 4 and 9.

```
SELECT *
FROM customer_purchases
WHERE product_id IN (4, 9);
```

--Write a query that returns all customer purchases and a new calculated column 'price'
(quantity ---* cost_to_customer_per_qty), filtered by vendor IDs between 8 and 10 (inclusive).

```
SELECT
  *,
  quantity * cost_to_customer_per_qty AS price
FROM customer_purchases
WHERE
  vendor_id BETWEEN 8 AND 10;
```

--CASE;1 ---Using the product table, write a query that outputs the product_id and
product_name columns and adds a column called prod_qty_type_condensed
---that displays "unit" if product_qty_type is "unit," and "bulk" otherwise.

```
 SELECT
  product_id,
  product_name,
  CASE
    WHEN product_qty_type = 'unit' THEN 'unit'
    ELSE 'bulk'
  END AS prod_qty_type_condensed
FROM product;
```

---Q Case2:

```
SELECT
 product_id,
 product_name,
 CASE
  WHEN product_qty_type = 'unit' THEN 'unit'
  ELSE 'bulk'
 END AS prod_qty_type_condensed,
 CASE
  WHEN product_name LIKE '%pepper%' THEN 1
  ELSE 0
 END AS pepper_flag
FROM product;
```

---JOIN---Write a query that INNER JOINs the vendor table to the vendor_booth_assignments table on the vendor_id field they both have in
--common, and sorts the result by vendor_name, then market_date.

```
SELECT
 T1.vendor_name,
 T2.market_date
FROM vendor AS T1
INNER JOIN vendor_booth_assignments AS T2
 ON T1.vendor_id = T2.vendor_id
ORDER BY
 T1.vendor_name,
 T2.market_date;
```

---Section 3: Advanced SQL
---AGGREGATE::Write a query that determines how many times each vendor has rented a booth at
----the farmer's market by counting the vendor booth assignments per vendor_id.

```
SELECT
 vendor_id,
 COUNT(vendor_id) AS booth_rentals
FROM vendor_booth_assignments
GROUP BY
 vendor_id;
```

--Write a query that generates a list of customers who have spent more
---than $2000 at the market, sorted by last name, then first name.

```
SELECT
  T1.customer_first_name,
  T1.customer_last_name
FROM customer AS T1
JOIN customer_purchases AS T2
  ON T1.customer_id = T2.customer_id
GROUP BY
  T1.customer_id
HAVING
  SUM(T2.quantity * T2.cost_to_customer_per_qty) > 2000
ORDER BY
  T1.customer_last_name,
  T1.customer_first_name;
```

---Temp Table
--Insert the original vendor table into a temp.new_vendor and then add a 10th vendor.

```
SELECT *

FROM vendor;

INSERT INTO temp.new_vendor (vendor_id, vendor_name, vendor_type,
vendor_owner_first_name, vendor_owner_last_name)

VALUES (10, 'KAUSHIK Superfood Store', 'Fresh Vegetable', 'Aman', 'Kauhsik');
```

---Check the status

```
SELECT *
FROM temp.new_vendor;
```

--- Lets try to deleted 10th vendor::

```
DELETE FROM temp.new_vendor
WHERE vendor_id = 10;

SELECT *
```

FROM temp.new_vendor;
---As We can see 10th vendor which was added recently has been deleted:::

---- again add it

```
SELECT *
FROM vendor;
INSERT INTO temp.new_vendor (vendor_id, vendor_name, vendor_type,
vendor_owner_first_name, vendor_owner_last_name)
```

VALUES (10, 'KAUSHIK Superfood Store', 'Fresh Vegetable', 'Aman', 'Kauhsik');

---check it gain ,,,wheather it has been added;;;

```
SELECT *
FROM temp.new_vendor;
```

---date:::

----Get the customer_id, month, and year (in separate columns) of every purchase in the customer_purchases table.

```
SELECT
  customer_id,
  STRFTIME('%m', market_date) AS month,
  STRFTIME('%Y', market_date) AS year
FROM customer_purchases;
```