PlacementDost Internship

First project: DA Restaurant Orders

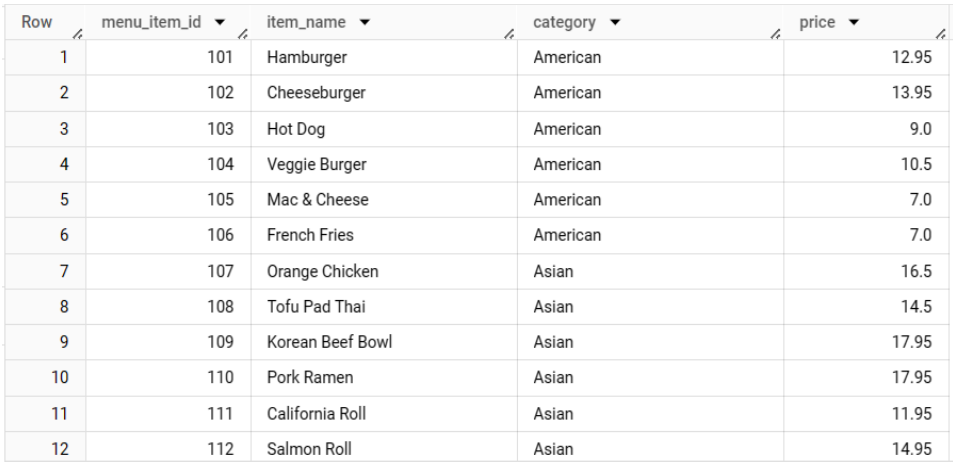
DA Restaurant Orders

1-using BigQuery, MySql

2-

- Retrieve all columns from the menu_items table.

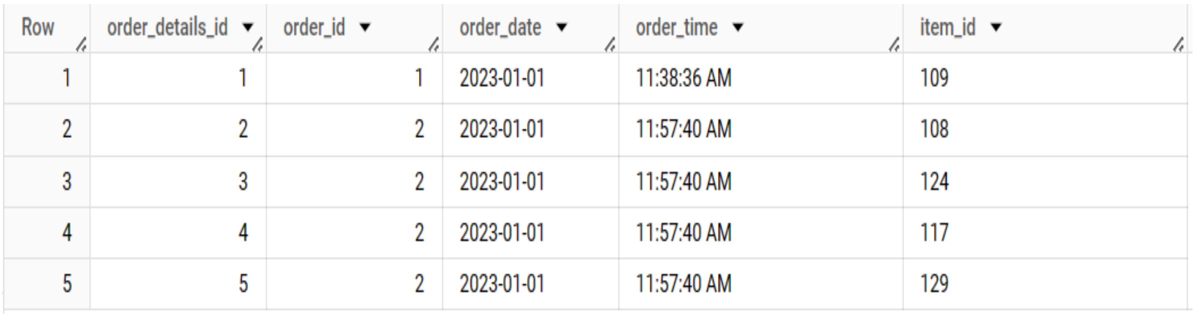
```
select
*
from menu_items
```



Row	menu_item_id	item_name	category	price
1	101	Hamburger	American	12.95
2	102	Cheeseburger	American	13.95
3	103	Hot Dog	American	9.0
4	104	Veggie Burger	American	10.5
5	105	Mac & Cheese	American	7.0
6	106	French Fries	American	7.0
7	107	Orange Chicken	Asian	16.5
8	108	Tofu Pad Thai	Asian	14.5
9	109	Korean Beef Bowl	Asian	17.95
10	110	Pork Ramen	Asian	17.95
11	111	California Roll	Asian	11.95
12	112	Salmon Roll	Asian	14.95

- Display the first 5 rows from the order_details table.

```
select
*
from order_details
limit 5
```

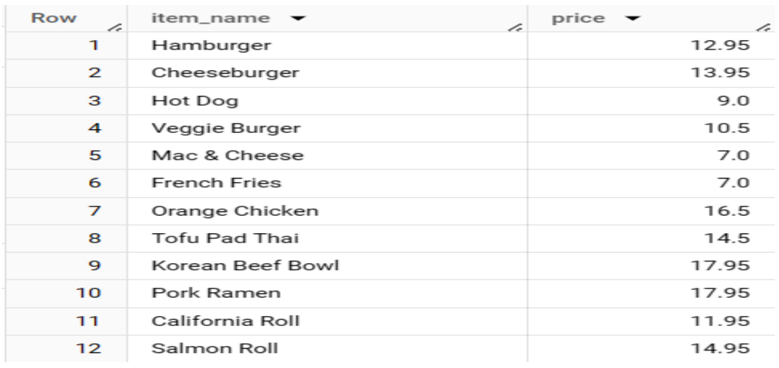


Row	order_details_id	order_id	order_date	order_time	item_id
1	1	1	2023-01-01	11:38:36 AM	109
2	2	2	2023-01-01	11:57:40 AM	108
3	3	2	2023-01-01	11:57:40 AM	124
4	4	2	2023-01-01	11:57:40 AM	117
5	5	2	2023-01-01	11:57:40 AM	129

3. Filtering and Sorting:

-Select the item_name and price columns for items in the 'Main Course' category.

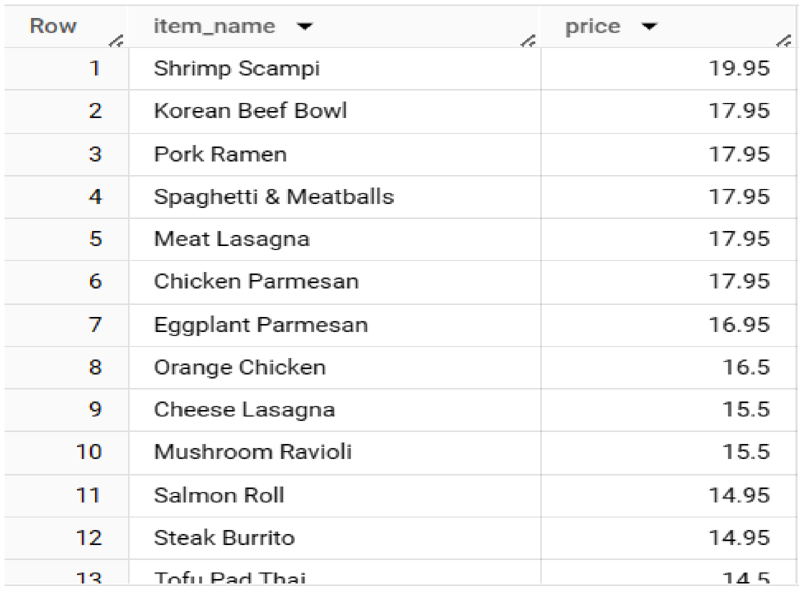
```
select
item_name , price
from menu_items
```



Row	item_name	price
1	Hamburger	12.95
2	Cheeseburger	13.95
3	Hot Dog	9.0
4	Veggie Burger	10.5
5	Mac & Cheese	7.0
6	French Fries	7.0
7	Orange Chicken	16.5
8	Tofu Pad Thai	14.5
9	Korean Beef Bowl	17.95
10	Pork Ramen	17.95
11	California Roll	11.95
12	Salmon Roll	14.95

- Sort the result by price in descending order

```
select
item_name , price
from menu_items
order by price desc
```

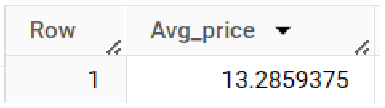


Row	item_name	price
1	Shrimp Scampi	19.95
2	Korean Beef Bowl	17.95
3	Pork Ramen	17.95
4	Spaghetti & Meatballs	17.95
5	Meat Lasagna	17.95
6	Chicken Parmesan	17.95
7	Eggplant Parmesan	16.95
8	Orange Chicken	16.5
9	Cheese Lasagna	15.5
10	Mushroom Ravioli	15.5
11	Salmon Roll	14.95
12	Steak Burrito	14.95
13	Tofu Pad Thai	14.5

4. Aggregate Functions:

- Calculate the average price of menu items.

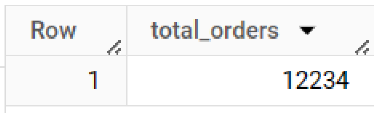
```
select
avg(price) as Avg_price
from menu_items
```



Row	Avg_price
1	13.2859375

- Find the total number of orders placed.

```
select
count(order_id) as total_orders
from order_details
```



Row	total_orders
1	12234

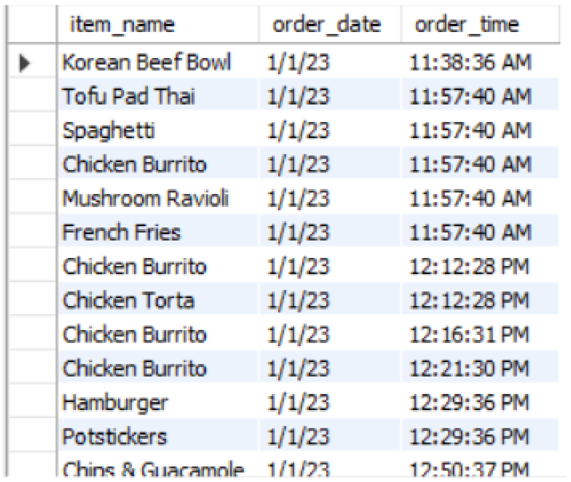
5. Joins:

- Retrieve the item_name, order_date, and order_time for all items in the order_details table, including their respective menu item details.

```
select item_name, order_date,  order_time

from order_details left join menu_items

 on order_details2.item_id=menu_items.menu_item_id
```

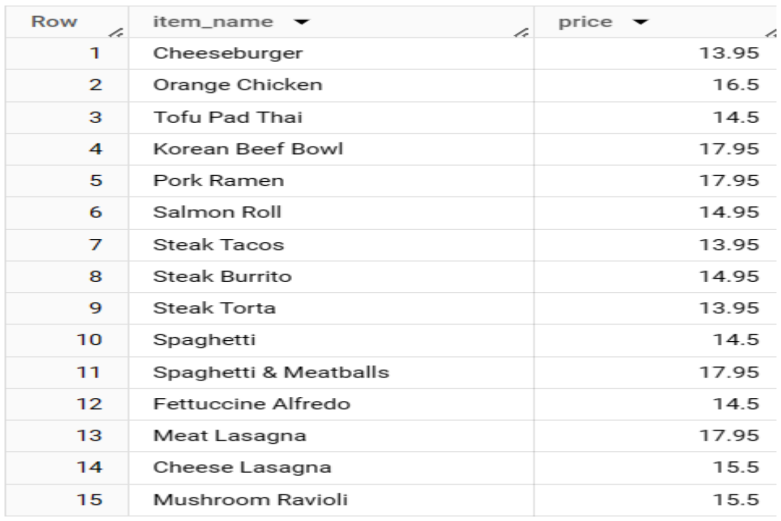


item_name	order_date	order_time
Korean Beef Bowl	1/1/23	11:38:36 AM
Tofu Pad Thai	1/1/23	11:57:40 AM
Spaghetti	1/1/23	11:57:40 AM
Chicken Burrito	1/1/23	11:57:40 AM
Mushroom Ravioli	1/1/23	11:57:40 AM
French Fries	1/1/23	11:57:40 AM
Chicken Burrito	1/1/23	12:12:28 PM
Chicken Torta	1/1/23	12:12:28 PM
Chicken Burrito	1/1/23	12:16:31 PM
Chicken Burrito	1/1/23	12:21:30 PM
Hamburger	1/1/23	12:29:36 PM
Potstickers	1/1/23	12:29:36 PM
Chips & Guacamole	1/1/23	12:50:37 PM

6. Subqueries:

- List the menu items (item_name) with a price greater than the average price of all menu items.

```
select
    item_name,price
from menu_items
where price >(select avg(price) from menu_items)
```

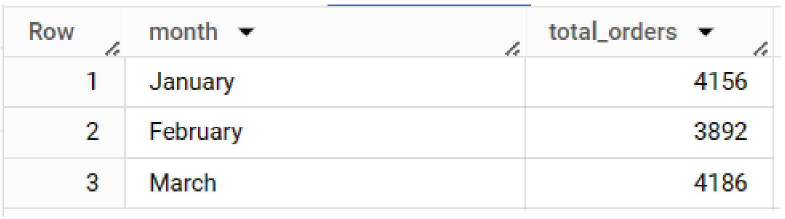


Row	item_name	price
1	Cheeseburger	13.95
2	Orange Chicken	16.5
3	Tofu Pad Thai	14.5
4	Korean Beef Bowl	17.95
5	Pork Ramen	17.95
6	Salmon Roll	14.95
7	Steak Tacos	13.95
8	Steak Burrito	14.95
9	Steak Torta	13.95
10	Spaghetti	14.5
11	Spaghetti & Meatballs	17.95
12	Fettuccine Alfredo	14.5
13	Meat Lasagna	17.95
14	Cheese Lasagna	15.5
15	Mushroom Ravioli	15.5

7. Date and Time Functions:

- Extract the month from the order_date and count the number of orders placed in each month.

```
select
    FORMAT_DATETIME("%B", DATETIME(order_date)) as
    month ,count(order_id) as total_orders
from order_details
group by month
```

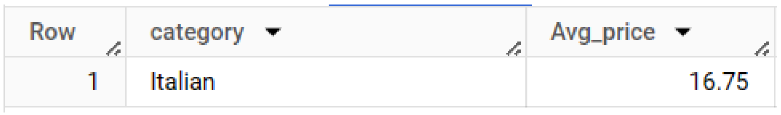


Row	month	total_orders
1	January	4156
2	February	3892
3	March	4186

8. Group By and Having:

- Show the categories with the average price greater than $15.

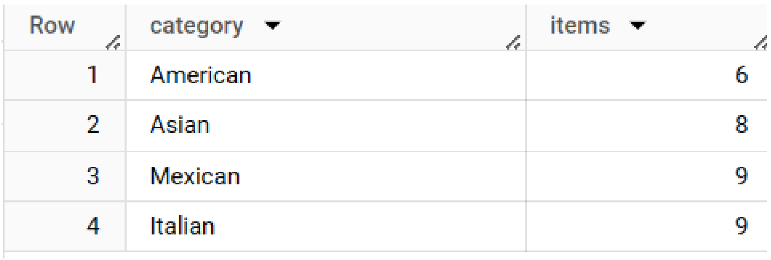
```
select
   category,avg(price) as Avg_price
from menu_items
group by category
having Avg_price >15
```



Row	category	Avg_price
1	Italian	16.75

- Include the count of items in each category.

```
select
   category,count(item_name) as items
from menu_items
group by category
```

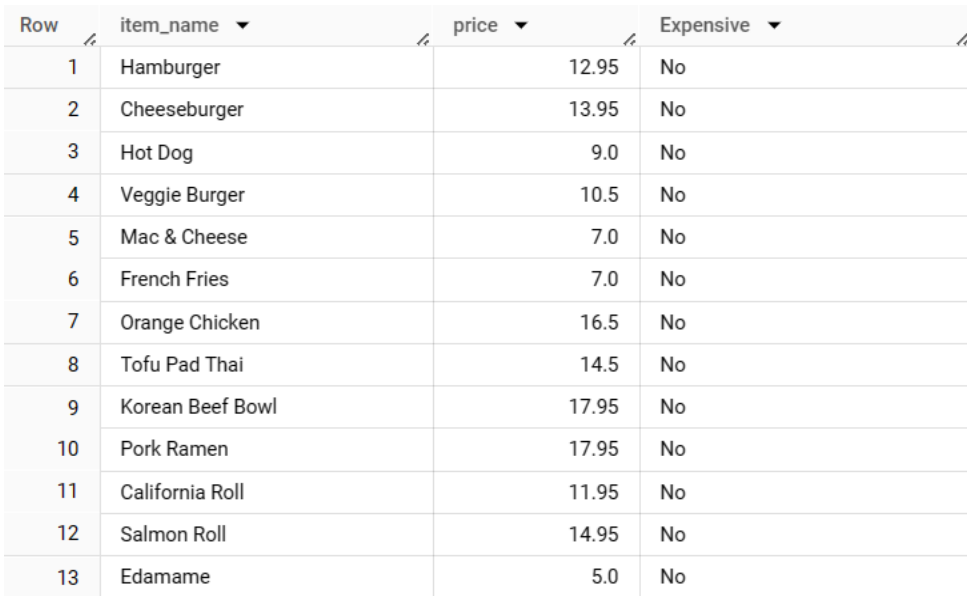


Row	category	items
1	American	6
2	Asian	8
3	Mexican	9
4	Italian	9

9. Conditional Statements:

- Display the item_name and price, and indicate if the item is priced above $20 with a new column named 'Expensive'.

```
SELECT item_name, price,
  CASE WHEN price > 20 THEN 'Yes' ELSE 'No' END AS Expensive
FROM menu_items;
```



Row	item_name	price	Expensive
1	Hamburger	12.95	No
2	Cheeseburger	13.95	No
3	Hot Dog	9.0	No
4	Veggie Burger	10.5	No
5	Mac & Cheese	7.0	No
6	French Fries	7.0	No
7	Orange Chicken	16.5	No
8	Tofu Pad Thai	14.5	No
9	Korean Beef Bowl	17.95	No
10	Pork Ramen	17.95	No
11	California Roll	11.95	No
12	Salmon Roll	14.95	No
13	Edamame	5.0	No

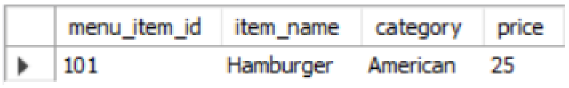
10. Data Modification

- Update: - Update the price of the menu item with item_id = 101 to $25.

```
update menu_items

set price=25

where menu_item_id = 101 ;
```

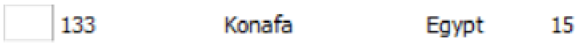


menu_item_id	item_name	category	price
101	Hamburger	American	25

11. Data Modification - Insert:

- Insert a new record into the menu_items table for a dessert item.

```
insert into menu_items values(133,"Konafa","Egypt",15)
```



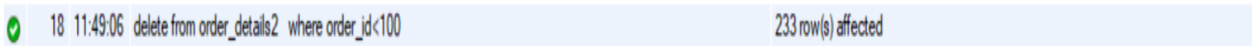
133	Konafa	Egypt	15

12. Data Modification - Delete:

- Delete all records from the order_details table where the order_id is less than 100.

```
delete from order_details

 where order_id < 100
```



```
18  11:49:06  delete from order_details2  where order_id<100    233 row(s) affected
```

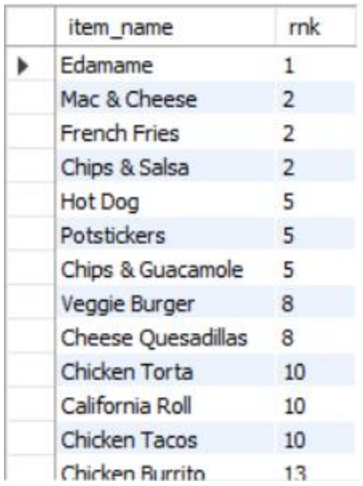
13. Window Functions - Rank:

- Rank menu items based on their prices, displaying the item_name and its rank.

```
 select m.item_name,

 rank() over(order by price) as rnk

 from menu_items m
```

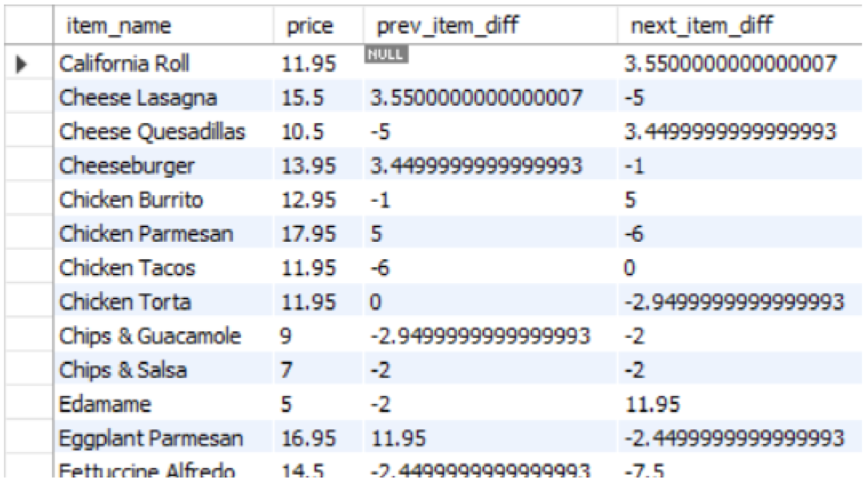


item_name	rnk
Edamame	1
Mac & Cheese	2
French Fries	2
Chips & Salsa	2
Hot Dog	5
Potstickers	5
Chips & Guacamole	5
Veggie Burger	8
Cheese Quesadillas	8
Chicken Torta	10
California Roll	10
Chicken Tacos	10
Chicken Burrito	13

14. Window Functions - Lag and Lead:

- Display the item_name and the price difference from the previous and next menu item.

```
select m.item_name ,m.price,
    price - lag(price) over (order by item_name) as prev_item_diff,
    lead(price) over(order by item_name) - price as next_item_diff
from menu_items m
```

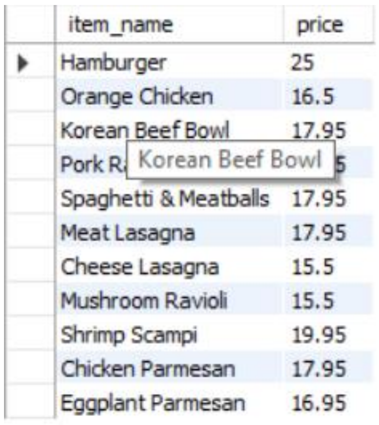


item_name	price	prev_item_diff	next_item_diff
California Roll	11.95	NULL	3.5500000000000007
Cheese Lasagna	15.5	3.5500000000000007	-5
Cheese Quesadillas	10.5	-5	3.4499999999999993
Cheeseburger	13.95	3.4499999999999993	-1
Chicken Burrito	12.95	-1	5
Chicken Parmesan	17.95	5	-6
Chicken Tacos	11.95	-6	0
Chicken Torta	11.95	0	-2.9499999999999993
Chips & Guacamole	9	-2.9499999999999993	-2
Chips & Salsa	7	-2	-2
Edamame	5	-2	11.95
Eggplant Parmesan	16.95	11.95	-2.4499999999999993
Fettuccine Alfredo	14.5	-2.4499999999999993	-7.5

15. Common Table Expressions (CTE):

- Create a CTE that lists menu items with prices above $15.

```
with cte as (
select item_name,price
from menu_items)
select * from cte
where price >15
```



	item_name	price
▶	Hamburger	25
	Orange Chicken	16.5
	Korean Beef Bowl	17.95
	Pork R	5
	Spaghetti & Meatballs	17.95
	Meat Lasagna	17.95
	Cheese Lasagna	15.5
	Mushroom Ravioli	15.5
	Shrimp Scampi	19.95
	Chicken Parmesan	17.95
	Eggplant Parmesan	16.95

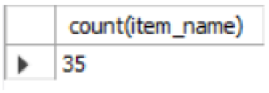
- Use the CTE to retrieve the count of such items.

```
with cte as (

select item_name,price

from menu_items)

select count(item_name) from cte
```



	count(item_name)
▶	35

16. Advanced Joins:

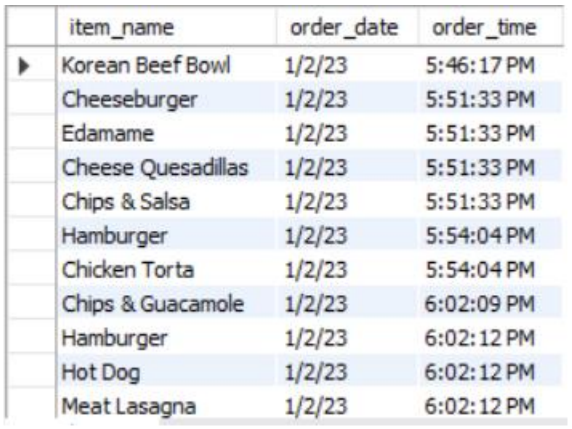
- Retrieve the order_id, item_name, and price for all orders with their respective menu item details.

- Include rows even if there is no matching menu item.

```
select item_name, order_date,  order_time

from order_details2  full join menu_items

 on item_id=menu_item_id
```



item_name	order_date	order_time
Korean Beef Bowl	1/2/23	5:46:17 PM
Cheeseburger	1/2/23	5:51:33 PM
Edamame	1/2/23	5:51:33 PM
Cheese Quesadillas	1/2/23	5:51:33 PM
Chips & Salsa	1/2/23	5:51:33 PM
Hamburger	1/2/23	5:54:04 PM
Chicken Torta	1/2/23	5:54:04 PM
Chips & Guacamole	1/2/23	6:02:09 PM
Hamburger	1/2/23	6:02:12 PM
Hot Dog	1/2/23	6:02:12 PM
Meat Lasagna	1/2/23	6:02:12 PM

17. Unpivot Data:

- Unpivot the menu_items table to show a list of menu item properties (item_id, item_name, category, price).

```
SELECT
  MAX(CASE WHEN col = 'menu_item_id' THEN value ELSE NULL END) AS
menu_item_id,
  MAX(CASE WHEN col = 'item_name' THEN value ELSE NULL END) AS
item_name,
  MAX(CASE WHEN col = 'category' THEN value ELSE NULL END) AS
category,
  MAX(CASE WHEN col = 'price' THEN value ELSE NULL END) AS price
FROM (
  SELECT  'menu_item_id' AS col, menu_item_id AS value FROM menu_items
  UNION ALL
  SELECT  'item_name' AS col, item_name AS value FROM menu_items
  UNION ALL
  SELECT  'category' AS col, category AS value FROM menu_items
  UNION ALL
  SELECT 'price' AS col, price AS value FROM menu_items
) AS unpivoted_data
;
```

18. Dynamic SQL:

- Write a dynamic SQL query that allows users to filter menu items based on category and price range.

```
SELECT
  menu_item_id,
  item_name,
  category,
  price
FROM menu_items
WHERE (category IS NULL OR category = @category)
  AND price BETWEEN @min_price AND @max_price;
```

19. Stored Procedure:

- Create a stored procedure that takes a menu category as input and returns the average price for that category.

```
DELIMITER //
CREATE PROCEDURE Avrage_Price(IN category VARCHAR(30))
BEGIN
  SELECT category,AVG(price) AS average_price
  FROM menu_items
  where category= @category;

END //

DELIMITER ;

CALL Avrage_Price('');
```

20. Triggers:

- Design a trigger that updates a log table whenever a new order is inserted into the order_details table.

```
CREATE TRIGGER order_log_trigger

AFTER INSERT ON order_details

FOR EACH ROW

BEGIN

INSERT INTO (order_details_id,order_id,order_date , order_time,
item_id) VALUES
(NEW.order_details_id,NEW.order_id,New.order_date,New.order_time,
NEW.item_id, NOW());

END;

[

CREATE DEFINER=`root`@`localhost` TRIGGER `order_log_trigger` AFTER
INSERT ON `order_details2` FOR EACH ROW

BEGIN

  INSERT INTO order_logs (order_details_id,order_id,order_date ,
order_time, item_id) VALUES
(NEW.order_details_id,NEW.order_id,New.order_date,New.order_time,
NEW.item_id, NOW());

END

]
```