Team 1 Co.

RESTAURANT MANAGEMENT PROJECT

Presented for :
ITI Data Analytics Team

Presented by :
Meran Ehab
Abdelrahman Khaled
Farrah Hatem
Adham Ayman
Nehal Saeed

TABLE OF CONTENTS

1. Introduction

1.1. Description:

Our task was to develop a robust Restaurant Management System aimed at optimizing restaurant operations, enhancing customer experience, and improving overall efficiency in managing orders, inventory, and staff. With a focus on streamlining processes and adhering to industry standards, our system aims to revolutionize how restaurants operate in the digital age.

1.2. Description:

Our project objectives were carefully crafted to address the core needs of restaurant management. From designing and implementing dimensional modeling to integrating analytics and reporting functionalities, our goals encompassed various aspects crucial for the success of our system. Key objectives include:

- Designing and implementing dimensional modeling for storing essential data such as menu items, customer orders, inventory, employee information, and sales data.
- Integrating features for inventory management, order tracking, and supplier information to ensure seamless operations.
- Incorporating analytics and reporting functionalities to analyze sales trends, monitor performance, and optimize operations.
- Ensuring data security, privacy, and compliance with industry regulations to maintain trust and credibility.

1.3. Business Requirements

The Restaurant Management System (RMS) project aims to develop a comprehensive solution for managing restaurant operations, enhancing customer experience, and optimizing efficiency in various aspects of restaurant management. This document outlines the key business requirements to be addressed by the RMS.

1. Menu Management:
   - Objective: Enable restaurant staff to efficiently manage menu items, their descriptions, prices, and availability status.
   - Requirements:
     - The RMS should provide functionality for creating, updating, and deleting menu items.
     - Each menu item should include attributes such as name, description, price, and availability status.
     - Menu items should be categorized by type (e.g., appetizers, entrees, desserts) for organized navigation.

2. Order Management:
   - Objective: Facilitate the efficient processing of customer orders and provide real-time updates on order status.
   - Requirements:
     - Staff members should be able to take orders, modify items, and manage order statuses (e.g., pending, in progress, completed).
     - Kitchen staff should have access to real-time updates on order status to prepare orders efficiently.

3. Reservation System:
   - Objective: Streamline the management of table reservations and provide flexibility in handling reservation requests.
   - Requirements:
     - The RMS should support the management of table reservations, capturing details such as date, time, party size, and special requests.
     - Staff should be able to view, modify, and cancel reservations as needed.

4. Seating Arrangement:
   - Objective: Optimize table management and seating arrangements for efficient restaurant operations.
   - Requirements:
     - The RMS should provide a visual representation of the restaurant layout and available seating options.
     - Staff should be able to assign tables to reservations and walk-in customers based on availability and customer preferences.

5. Inventory Control:
   - Objective: Effectively monitor and manage inventory levels to ensure adequate stock for restaurant operations.
   - Requirements:
     - The system should track inventory levels for ingredients, beverages, and other supplies.
     - Staff should receive notifications for low stock levels and be able to automate replenishment orders with suppliers.

6. Employee Management:
- Objective: Streamline employee scheduling, performance management, and communication within the restaurant.
- Requirements:
  - The RMS should maintain employee profiles, including roles, contact information, and work schedules.
  - Staff assignments, shifts, and performance evaluations should be managed within the system.

7. Reporting and Analytics:
- Objective: Provide insights into restaurant performance, customer preferences, and operational efficiency.
- Requirements:
  - The system should generate reports on sales performance, revenue trends, and customer feedback.
  - Data analysis should identify popular menu items, peak hours, and opportunities for upselling.

1.4. Problem Definition:

1. Lack Of Data Existence For Restaurants
- Problem: The absence of comprehensive and reliable data for restaurants hinders the development of effective restaurant management systems. Without access to relevant data, decision-making and system functionalities are compromised.

- Impact: Inability to make data-driven decisions, limited functionality in restaurant management systems, reduced competitiveness, and growth opportunities.

- Solution Approach: Conduct thorough data collection efforts, establish partnerships with data providers, implement data enrichment techniques, and develop scalable data management infrastructure.

2. Limited Features of Found Data
- Problem: Found data for restaurant management systems often lacks the breadth and depth needed to support comprehensive functionality, leading to gaps in insights and system capabilities.

- Impact: Incomplete and inaccurate insights, inefficient system functionalities, and increased development complexity and costs.

- Solution Approach: Conduct data quality assessments, implement data enrichment strategies, leverage advanced analytics techniques, and establish data partnerships and collaborations.

3. Integrate Application & Database

- Problem: Inefficient integration between the application layer and the database in restaurant management systems results in data inconsistency, performance degradation, and scalability limitations.

- Impact: Data inconsistency and integrity issues, performance degradation and latency, and scalability constraints.

- Solution Approach: Adopt robust integration technologies, ensure data consistency and synchronization, optimize data access and retrieval, and design for scalability and flexibility.

1.5 Tools & Technologies

Tools:

- Kaggle
- Informatica PowerCenter
- Toad
- Power BI
- Jupyter Notebook
- Google Collab
- Visual Studio Code
- PowerQuery
- DAX

Technologies:

- SQL
- PL/SQL
- Python
- ETL
- Analytical SQL

2. Data Collection:

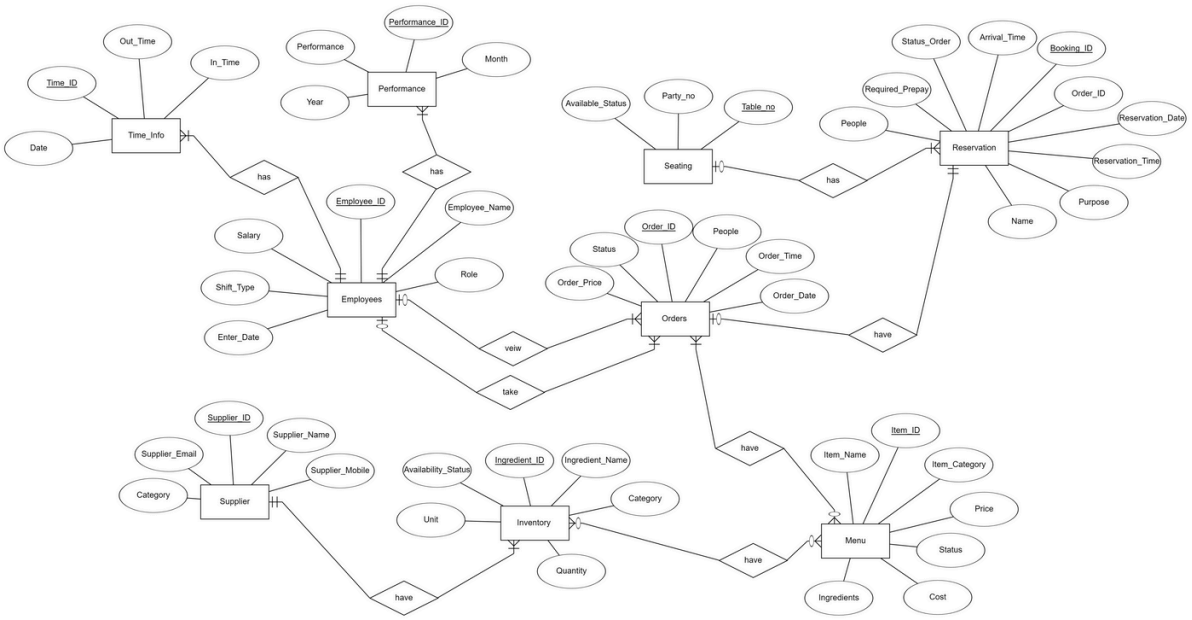
2.1. Data Sources

GitHub

2.2. Tables

- Orders
- Order Items
- Reservations
- Inventory
- Menu
- Item Ingredient
- Employee Info
- Employees In & Out Times
- Employees Performance
- Seating
- Supplier

2.3. ERD



2.4. Mapping

- **Orders**
(#ORDER_ID, ORDER_DATE, ORDER_TIME, Order_PRICE, PEOPLE, STATUS, employee_ID(fk), kitchen_Staff(fk))

- **Menu**
(#Item_ID, Item_Name, Item_Size, Item_Category, Ingredients, Price,Cost, Status)

- **Inventory**
(#Ingredient_ID, Ingredient_name, Category, Quantity, Unit, Availability Status, Supplier_id(fk))

- **Suppliers**
(#Supplier_id, Supplier_name, Supplier_mobile, Supplier_email, Category)

-**Reservation**
(#BOOKING_ID, Name, Reservation_DATE, Reservation_TIME, Arrival_Time, PEOPLE,REQUIRED_PREPAY,PURPOSE,STATUS_ORDER,TABLE_NUMBER(fk), ORDER_ID(fk))

- **Seating**(#table_no, party_no, avaliable_status)

- **Employees**(#Employee_ID, Employee_Name, Role, Shift_Type, Enter Date, Salary)

- **Performance**(#Performance_Id , Employee_Id(fk) , Month , Year, Performance)

- **Time_info**(#Time_Id , Employee_Id(fk),Date, In_time, Out_time)

- **order_Items**(#order_id(fk), #item_id(fk), quantity)

- **item_ingredients**(#Item_ID(fk), #Ingredient_ID(fk), Status)

3. Data Manipulation

3.1. PL/SQL SCRIPTS:

1. Trigger to assign tables to 'Dine in without Reservation'

```
create or replace trigger update_orders_and_seating
before insert or update on orders for each row
declare
   v_reserved_table_no reservation.table_number%type;
   v_next_reservation_exists number;
   v_90_minutes_before timestamp;
   v_90_minutes_after timestamp;
   v_available_table_no number;
   v_order_timestamp timestamp;
begin
   if :new.status = 'dine in without reservation' then
    v_order_timestamp := to_timestamp(to_char(:new.order_date, 'yyyy-mm-dd') || ' ' || to_char(:new.order_time, 'hh24:mi:ss'), 'yyyy-mm-dd hh24:mi:ss');

     v_90_minutes_before := v_order_timestamp - interval '90' minute;
     v_90_minutes_after := v_order_timestamp + interval '90' minute;

     begin
        select table_number
        into v_reserved_table_no
        from reservation
        where reservation_date = trunc(:new.order_date)
        and reservation_time >= v_90_minutes_before
        and reservation_time <= v_90_minutes_after
        and table_number = :new.table_no
        and rownum = 1;
```

```
begin
    if :new.status = 'dine in without reservation' then
      v_order_timestamp := to_timestamp(to_char(:new.order_date, 'yyyy-mm-dd') || ' ' || to_char(:new.order_time, 'hh24:mi:ss'), 'yyyy-mm-dd hh24:mi:ss');

      v_90_minutes_before := v_order_timestamp - interval '90' minute;
      v_90_minutes_after := v_order_timestamp + interval '90' minute;

      begin
        select table_number
        into v_reserved_table_no
        from reservation
        where reservation_date = trunc(:new.order_date)
        and reservation_time >= v_90_minutes_before
        and reservation_time <= v_90_minutes_after
        and table_number = :new.table_no
        and rownum = 1;
      exception
        when no_data_found then
          v_reserved_table_no := null;
      end;
```

```
    if v_reserved_table_no is null then

      if :new.table_no is null then
        select table_no
        into v_available_table_no
        from seating
        where party_no >= :new.people
        and available_status = 'true'
        and rownum = 1;

        if v_available_table_no is not null then
          :new.table_no := v_available_table_no;
          update seating
          set available_status = 'false'
          where table_no = v_available_table_no;
        else
          raise_application_error(-20001, 'no suitable table available');
          null;
        end if;
      end if;
    end if;
  end if;
end;
```

2. Trigger to assign tables to ‘Dine in with Reservation’

```
create or replace trigger assign_table_number
before insert on reservation
for each row
declare
  v_available_table_number seating.table_no%type;
begin
  select table_no into v_available_table_number
  from (
    select table_no, row_number() over (order by table_no) as rn
    from seating
    where available_status = 'true'
    and table_no not in (
      select reservation.table_number
      from reservation
      where trunc(reservation_date) = trunc(to_date(:new.reservation_date, 'yyyy-mm-dd'))
      and (
        reservation_time = :new.reservation_time or
        reservation_time between :new.reservation_time - interval '90' minute and :new.reservation_time + interval '90' minute
      )
    )
  )
  where rn = 1;
```

```
    update seating
    set available_status = 'false'
    where table_no = v_available_table_number;

exception
    when no_data_found then
        raise_application_error(-20001, 'no available table for the reservation at the specified time.');
    when others then
        raise_application_error(-20002, 'an error occurred while assigning table.');
end;
```

3. Trigger to check REQUIRED PREPAY

```
create or update trigger  update_required_prepay_trigger
before insert or update on RESERVATION for each row
begin
    if :new.people <= 10 then
        :new.required_prepay := 'true';
    end if;
end;
/
```

4. Procedure to Update Menu Items Availability

```
create or replace procedure update_menu_availability as
begin

    update item_ingredients
    set status = 'unavailable'
    where ingredient_id in (select id from inventory where quantity = 0);

    update menu
    set status = 'unavailable'
    where item_id in (select item_id from item_ingredients where ingredient_id in (select id from inventory where quantity = 0));

    dbms_output.put_line('Availability and status updated successfully.');
exception
    when others then
        rollback;
        dbms_output.put_line('an error occurred: ' || sqlerrm);
end update_menu_availability;
/
```

5. Procedure & Trigger before inserting order items

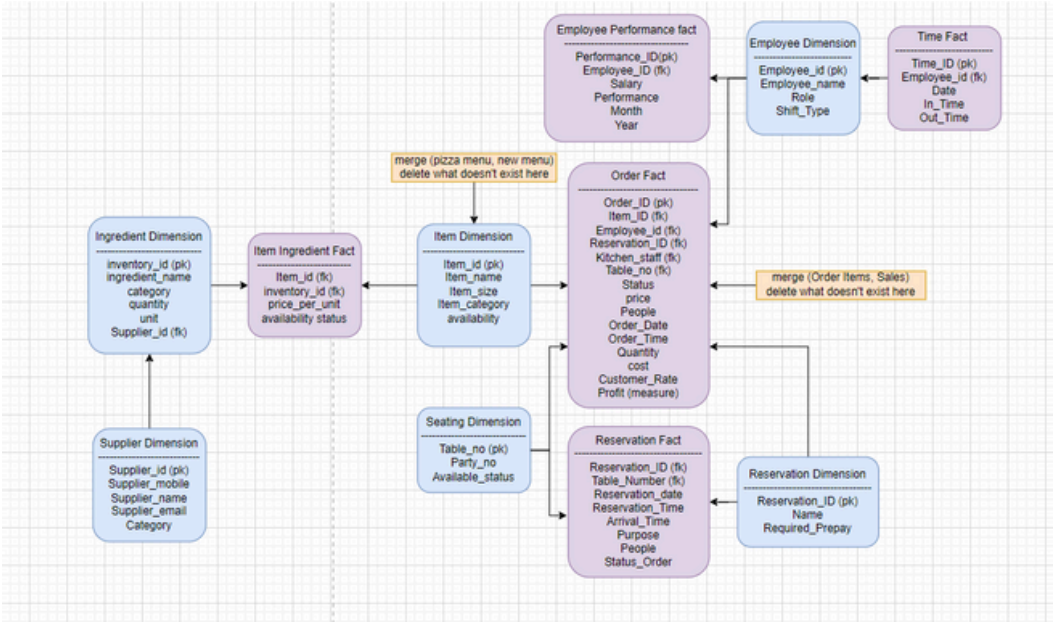
```
create or replace procedure check_item_availability (
    p_item_id in order_items.item_id%type,
    p_availability out varchar2
)
is
    item_status menu.status%type;
begin
    select status
    into item_status
    from menu
    where item_id = p_item_id;

    if item_status <> 'available' then
        p_availability := 'unavailable';
    else
        p_availability := 'available';
    end if;
end check_item_availability;
/
-------------------------------------------
create or replace trigger before_insert_order_items
before insert on order_items
for each row
declare
    v_item_availability varchar2(20);
begin
    check_item_availability(:new.item_id, v_item_availability);

    if v_item_availability = 'unavailable' then
        raise_application_error(-20001, 'the item is unavailable for ordering.');
    end if;
end;
/
```

6. JOB to reset the seating table

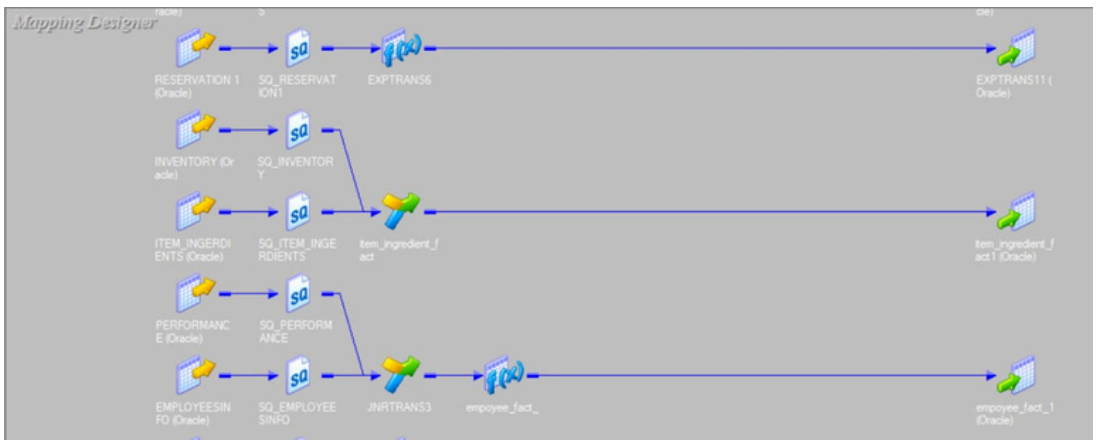
```
create or replace procedure update_seating_availability AS
begin
    update Seating
    set available_status = 'TRUE';
end;
/

begin
    dbms_scheduler.create_job(
        job_name => 'upd_seat_avail_job',
        job_type => 'PLSQL_BLOCK',
        job_action => 'begin update_seating_availability; end;',
        start_date => systimestamp + interval '1' hour,
        repeat_interval => 'freq = hourly',
        enabled => true,
        auto_drop => true
    );
end;
/
```

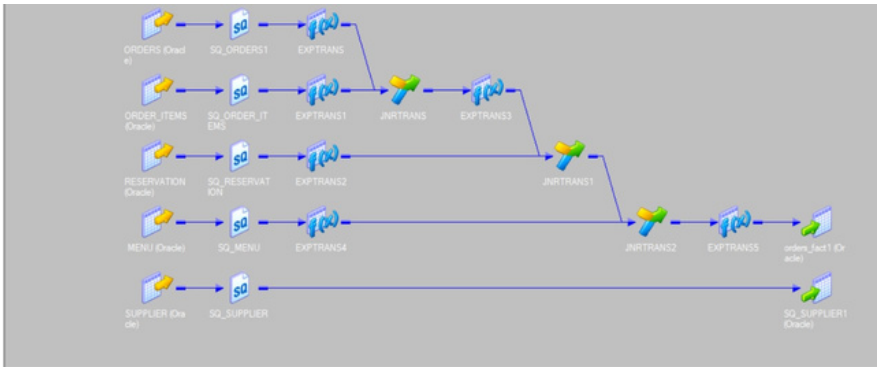
4. Dimensional Modeling

4.1. Modeling



4.1. Informatica Model





5. Analysis & Reports

5.1. BI Queries

- The Top 2 sold items per quarter of every year by Order

```
WITH OrderCounts AS (
   SELECT   item_id,
       TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'Q') AS quarter,
      TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY') AS year,
      COUNT(order_id) AS order_count
   FROM  orders_fact
   GROUP BY  item_id, TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY'),
TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'Q')
)
SELECT
   item_id, quarter, year, order_count
FROM
   ( SELECT  item_id, quarter, year order_count,
     ROW_NUMBER() OVER (PARTITION BY year, quarter ORDER BY
order_count DESC) AS row_num
      FROM  OrderCounts
   ) ranked
WHERE  row_num <= 2
ORDER BY  year, quarter;
```

Query Result

SQL | All Rows Fetched: 16 in 0.439 seconds

	ITEM_ID	QUARTER	YEAR	ORDER_COUNT
1	big meat s	1	2015	456
2	five cheese l	1	2015	367
3	big meat s	2	2015	446
4	five cheese l	2	2015	339
5	big meat s	3	2015	458
6	five cheese l	3	2015	357
7	big meat s	4	2015	451
8	thai ckn l	4	2015	354
9	ckn pasta	1	2016	905
10	veg pasta	1	2016	832
11	veg pasta	2	2016	909
12	ckn pasta	2	2016	825
13	veg pasta	3	2016	901
14	ckn pasta	3	2016	872
15	veg pasta	4	2016	882
16	ckn pasta	4	2016	848

- Total Sales Per Quarter Of Every Year

```
SELECT
 DISTINCT TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'Q') AS quarter,
 TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY') as year,
 SUM(quantity * price) OVER (PARTITION BY TO_CHAR(TO_DATE(order_date,
'DD-MM-YY'), 'YYYY'),TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'Q')) AS
total_sum
FROM orders_fact
ORDER BY quarter;
```

Query Result

SQL | All Rows Fetched: 8 in 0.353 seconds

	QUARTER	YEAR	TOTAL_SUM
1	1	2015	1020500
2	1	2016	3998217.75
3	2	2015	1035948.75
4	2	2016	3955346.75
5	3	2015	1018281
6	3	2016	4025539
7	4	2015	990070.5
8	4	2016	3964010.75

- The Purpose People Come To Most (How Perceived At Market)

```
select purpose,count(purpose) as count
from reservation_fact
group by purpose
order by count(purpose) desc;
```

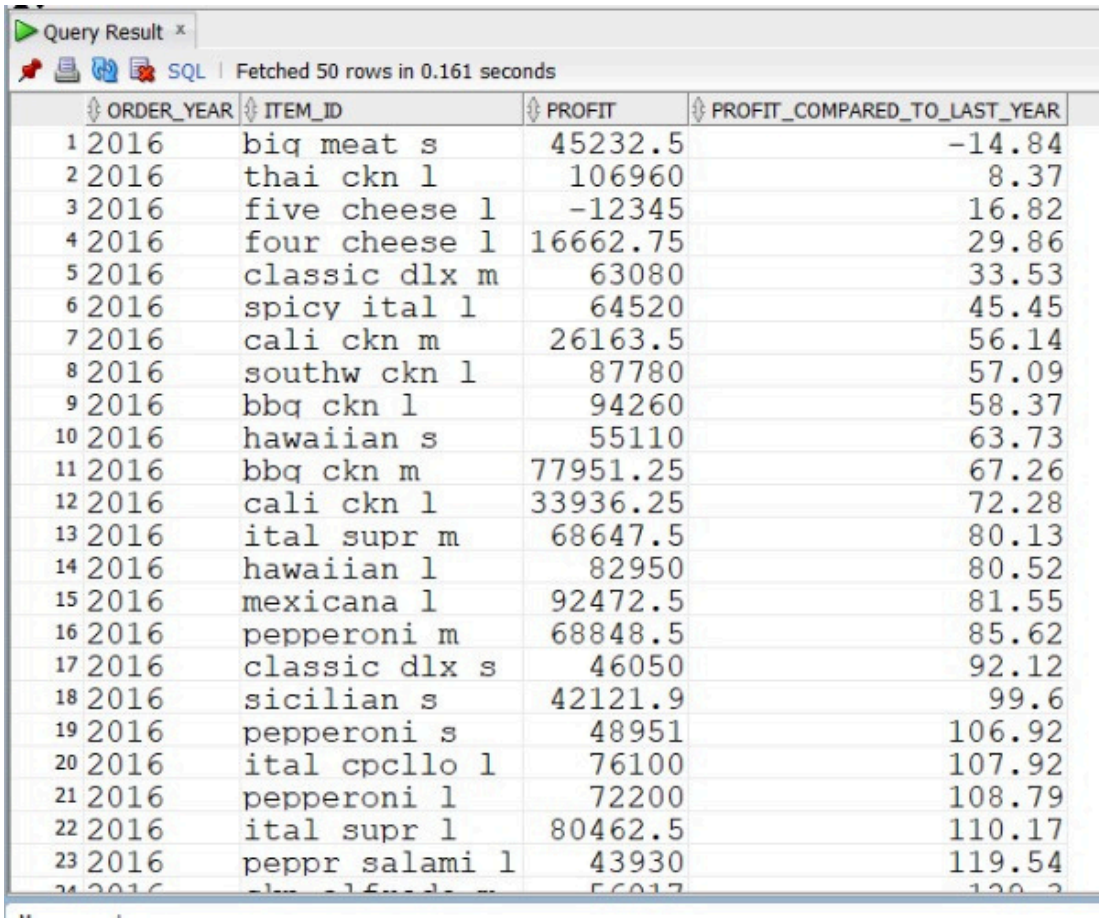
Query Result

SQL | All Rows Fetched: 9 in 0.25 seconds

	PURPOSE	COUNT
1	Dinner with Friends	12972
2	Single Dinner	7293
3	Sweet Anniversary	6288
4	Birthday Celebration	460
5	Important Date	412
6	Business Dinner	81
7	Family Gathering	29
8	Anniversary Celebration	20
9	Other	5

- Comparing The Profit Of Each Item by Year, Showing the percentage change.

```
WITH Profits AS (
 SELECT
 TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY') AS Order_Year,
 item_id, SUM(quantity * (price - cost)) AS Profit
 FROM orders_fact
 GROUP BY TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY'), item_id
),
PreviousYearProfit AS (
 SELECT item_id,Profit AS Current_Year_Profit,
 LAG(Profit) OVER (PARTITION BY item_id ORDER BY Order_Year) AS
Previous_Year_Profit,
 Order_Year
 FROM Profits
)
SELECT p.Order_Year, p.item_id, p.Profit,
 ROUND(((p.Profit - py.Previous_Year_Profit) * 100 / py.Previous_Year_Profit), 2)
AS Profit_Compared_To_Last_Year
FROM Profits p
LEFT JOIN PreviousYearProfit py
ON p.item_id = py.item_id AND p.Order_Year = py.Order_Year
WHERE p.Order_Year = 2016
ORDER BY Profit_Compared_To_Last_Year ;
```



Query Result

SQL | Fetched 50 rows in 0.161 seconds

	ORDER_YEAR	ITEM_ID	PROFIT	PROFIT_COMPARED_TO_LAST_YEAR
1	2016	big meat s	45232.5	-14.84
2	2016	thai ckn l	106960	8.37
3	2016	five cheese l	-12345	16.82
4	2016	four cheese l	16662.75	29.86
5	2016	classic dlx m	63080	33.53
6	2016	spicy ital l	64520	45.45
7	2016	cali ckn m	26163.5	56.14
8	2016	southw ckn l	87780	57.09
9	2016	bbq ckn l	94260	58.37
10	2016	hawaiian s	55110	63.73
11	2016	bbq ckn m	77951.25	67.26
12	2016	cali ckn l	33936.25	72.28
13	2016	ital supr m	68647.5	80.13
14	2016	hawaiian l	82950	80.52
15	2016	mexicana l	92472.5	81.55
16	2016	pepperoni m	68848.5	85.62
17	2016	classic dlx s	46050	92.12
18	2016	sicilian s	42121.9	99.6
19	2016	pepperoni s	48951	106.92
20	2016	ital cpcllo l	76100	107.92
21	2016	pepperoni l	72200	108.79
22	2016	ital supr l	80462.5	110.17
23	2016	peppr salami l	43930	119.54

- Cancellation Rate

```
WITH OrderCounts AS (
 SELECT COUNT(*) AS total_orders,
 COUNT(CASE WHEN status_order = 'Cancelled' THEN 1 END) AS
canceled_orders
 FROM reservation_fact
)
SELECT canceled_orders, total_orders, round((canceled_orders / total_orders)
* 100,2) AS cancellation_rate
FROM OrderCounts;
```

Query Result

SQL | All Rows Fetched: 1 in 0.077 seconds

	CANCELED_ORDERS	TOTAL_ORDERS	CANCELLATION_RATE
1	1797	27560	6.52

- Ranked Rush Hours For Reservation by Year

```
WITH HourlyReservationCounts AS (
 SELECT
 TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY') AS year,
 SUBSTR(order_time, 1, INSTR(order_time, ':') - 1) || SUBSTR(order_time,
INSTR(order_time, ' ') + 1) AS hour,
 COUNT(booking_id) AS number_of_reservations,
 ROW_NUMBER() OVER (PARTITION BY TO_CHAR(TO_DATE(order_date, 'DD-
MM-YY'), 'YYYY') ORDER BY COUNT(booking_id) DESC) AS hour_rank
 FROM orders_fact
 GROUP BY
 TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY'),
 SUBSTR(order_time, 1, INSTR(order_time, ':') - 1) || SUBSTR(order_time,
INSTR(order_time, ' ') + 1)
)
SELECT year, hour, number_of_reservations
FROM HourlyReservationCounts
WHERE hour_rank <= 3
ORDER BY year, hour_rank;
```

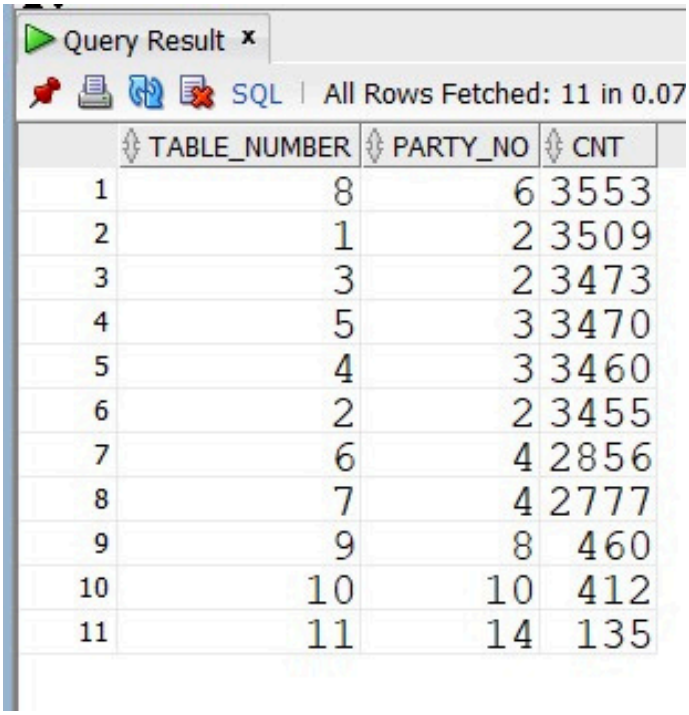
Query Result

SQL | All Rows Fetched: 6 in 0.142 seconds

	YEAR	HOUR	NUMBER_OF_RESERVATIONS
1	2015	12PM	3268
2	2015	1PM	3247
3	2015	6PM	2616
4	2016	4PM	3996
5	2016	12PM	3862
6	2016	3PM	3787

- No. Of Reservations For Each Table

```
SELECT distinct table_number, party_no , COUNT(*)over(partition by
table_number)as cnt
FROM reservation_fact , seating_dim
where table_number = table_no
order by cnt desc;
```



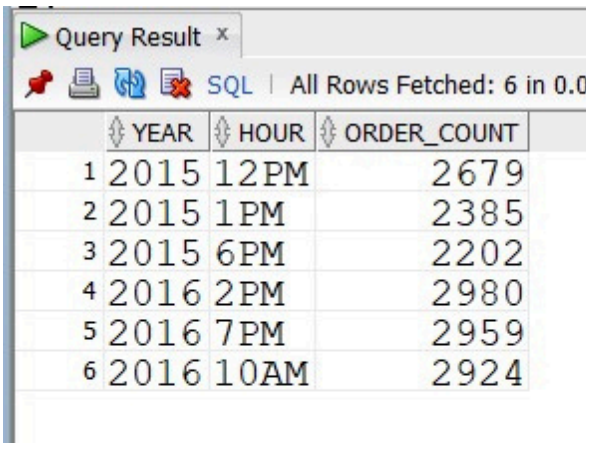
Query Result

SQL | All Rows Fetched: 11 in 0.07

	TABLE_NUMBER	PARTY_NO	CNT
1	8	6	3553
2	1	2	3509
3	3	2	3473
4	5	3	3470
5	4	3	3460
6	2	2	3455
7	6	4	2856
8	7	4	2777
9	9	8	460
10	10	10	412
11	11	14	135

- Most Take Away Orders Peak Hours

```
WITH HourlyOrderCounts AS (
 SELECT
 TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY') AS year,
 SUBSTR(order_time, 1, INSTR(order_time, ':') - 1) || SUBSTR(order_time,
INSTR(order_time, ' ') + 1) AS hour,
 COUNT(order_id) AS order_count,
 ROW_NUMBER() OVER (PARTITION BY TO_CHAR(TO_DATE(order_date, 'DD-
MM-YY'), 'YYYY') ORDER BY COUNT(order_id) DESC) AS hour_rank
 FROM orders_fact
 where STATUS = 'Take away'
 GROUP BY
 TO_CHAR(TO_DATE(order_date, 'DD-MM-YY'), 'YYYY'),
 SUBSTR(order_time, 1, INSTR(order_time, ':') - 1) || SUBSTR(order_time,
INSTR(order_time, ' ') + 1)

)
SELECT year, hour, order_count
FROM HourlyOrderCounts
WHERE hour_rank <= 3
ORDER BY year, hour_rank;
```



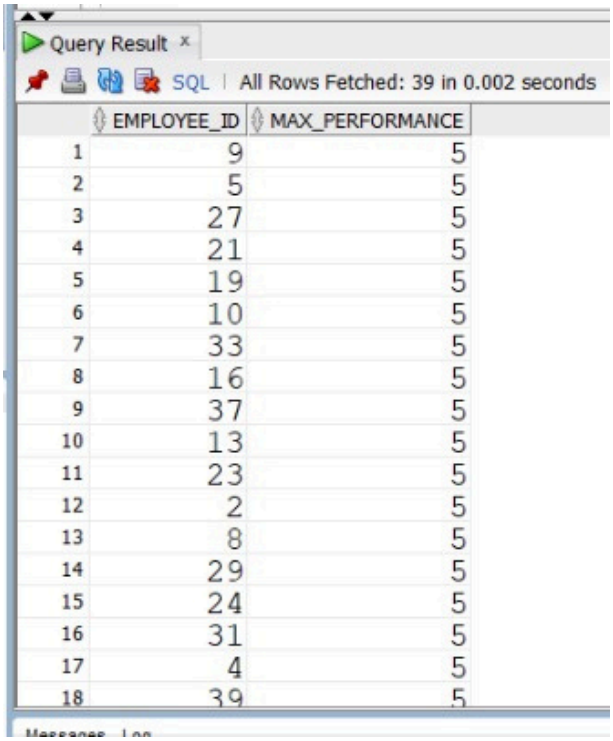
Query Result

SQL | All Rows Fetched: 6 in 0.0

	YEAR	HOUR	ORDER_COUNT
1	2015	12PM	2679
2	2015	1PM	2385
3	2015	6PM	2202
4	2016	2PM	2980
5	2016	7PM	2959
6	2016	10AM	2924

- Top Performer

```
SELECT employee_id, MAX(performance) AS max_performance
FROM empoyee_fact
GROUP BY employee_id
ORDER BY max_performance DESC;
```



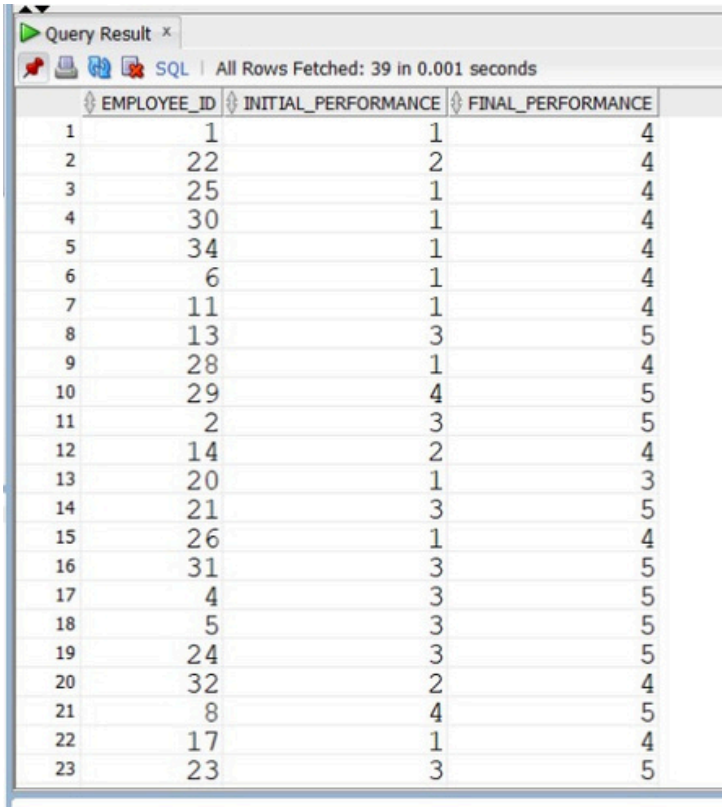
Query Result

SQL | All Rows Fetched: 39 in 0.002 seconds

	EMPLOYEE_ID	MAX_PERFORMANCE
1	9	5
2	5	5
3	27	5
4	21	5
5	19	5
6	10	5
7	33	5
8	16	5
9	37	5
10	13	5
11	23	5
12	2	5
13	8	5
14	29	5
15	24	5
16	31	5
17	4	5
18	39	5

- Performance Improvement

```
SELECT employee_id, MIN(performance) AS initial_performance,
MAX(performance) AS final_performance
FROM empoyee_fact
GROUP BY employee_id
HAVING MAX(performance) > MIN(performance);
```



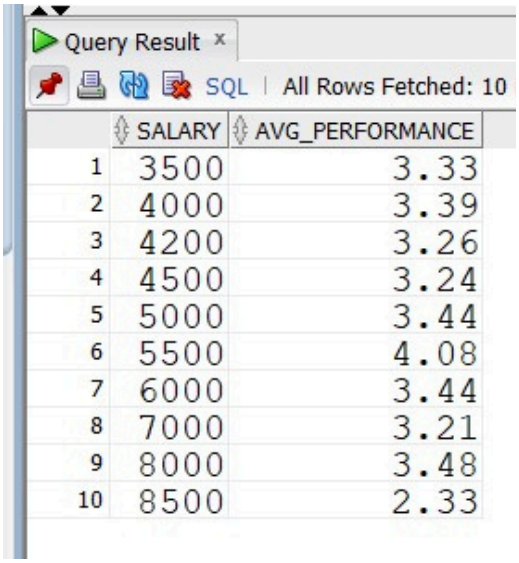
Query Result

SQL | All Rows Fetched: 39 in 0.001 seconds

	EMPLOYEE_ID	INITIAL_PERFORMANCE	FINAL_PERFORMANCE
1	1	1	4
2	22	2	4
3	25	1	4
4	30	1	4
5	34	1	4
6	6	1	4
7	11	1	4
8	13	3	5
9	28	1	4
10	29	4	5
11	2	3	5
12	14	2	4
13	20	1	3
14	21	3	5
15	26	1	4
16	31	3	5
17	4	3	5
18	5	3	5
19	24	3	5
20	32	2	4
21	8	4	5
22	17	1	4
23	23	3	5

- Salary Performance Correlation

```
SELECT salary, round(AVG(performance),2) AS avg_performance
FROM empoyee_fact
GROUP BY salary
ORDER BY salary;
```

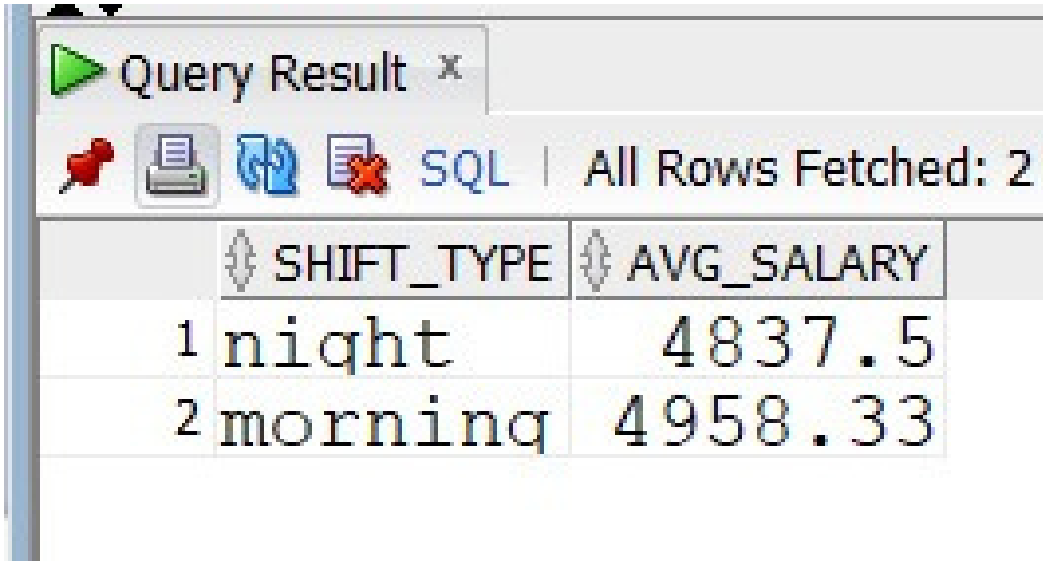

Query Result

SQL | All Rows Fetched: 10

	SALARY	AVG_PERFORMANCE
1	3500	3.33
2	4000	3.39
3	4200	3.26
4	4500	3.24
5	5000	3.44
6	5500	4.08
7	6000	3.44
8	7000	3.21
9	8000	3.48
10	8500	2.33

- Average Salary by Shift Time

```
SELECT SHIFT_TYPE, ROUND(AVG(SALARY),2) AS avg_salary
FROM EMPLOYEE_DIM,EMPOYEE_FACT
WHERE EMPLOYEE_DIM.employee_id = EMPOYEE_FACT.employee_id
GROUP BY SHIFT_TYPE;
```

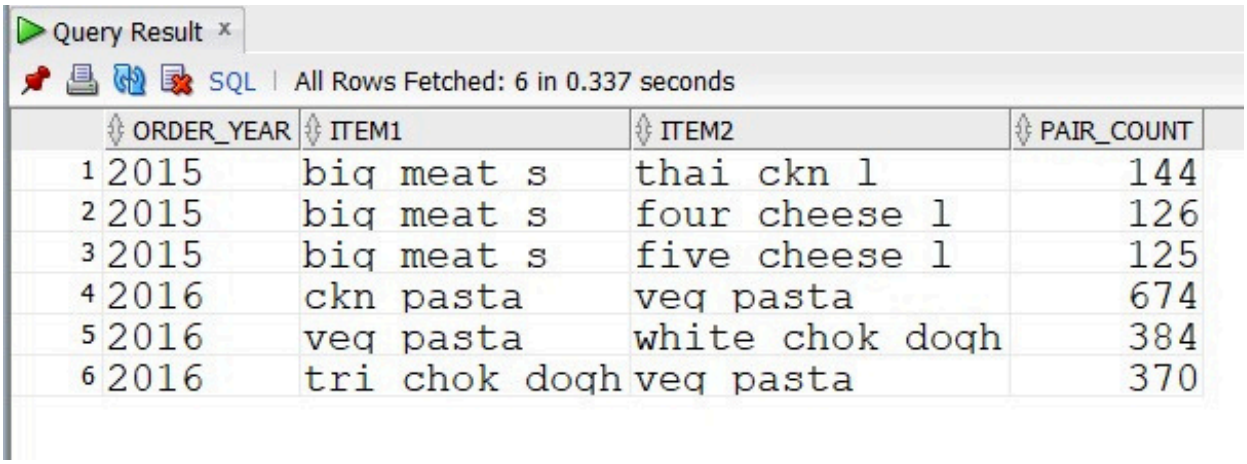

Query Result

SQL | All Rows Fetched: 2

	SHIFT_TYPE	AVG_SALARY
1	night	4837.5
2	morning	4958.33

- The Frequently Bought Together 3 Items by Year

```
WITH ItemPairs AS (
 SELECT
 TO_CHAR(TO_DATE(o1.order_date, 'DD-MM-YY'), 'YYYY') AS Order_Year,
 o1.item_id AS Item1, o2.item_id AS Item2, COUNT(*) AS Pair_Count
 FROM orders_fact o1
 INNER JOIN orders_fact o2
 ON o1.order_id = o2.order_id AND o1.item_id < o2.item_id
 GROUP BY TO_CHAR(TO_DATE(o1.order_date, 'DD-MM-YY'), 'YYYY'),
o1.item_id, o2.item_id
),
RankedPairs AS (
 SELECT Order_Year, Item1, Item2, Pair_Count,
 ROW_NUMBER() OVER (PARTITION BY Order_Year ORDER BY Pair_Count
DESC) AS Rank
 FROM ItemPairs
)
SELECT Order_Year, Item1, Item2, Pair_Count
FROM RankedPairs
WHERE Rank <=3 ;
```

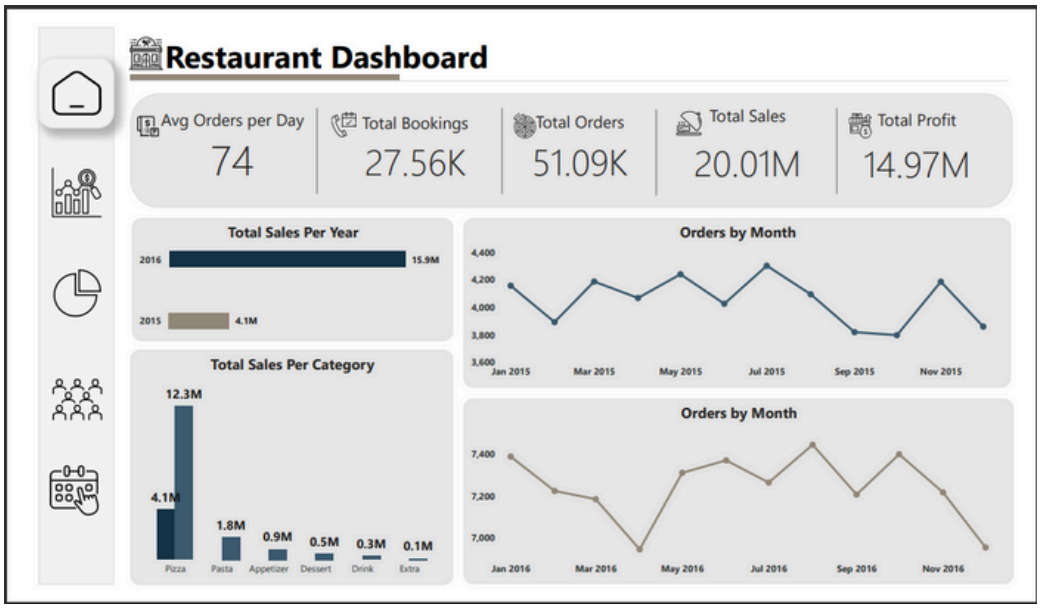


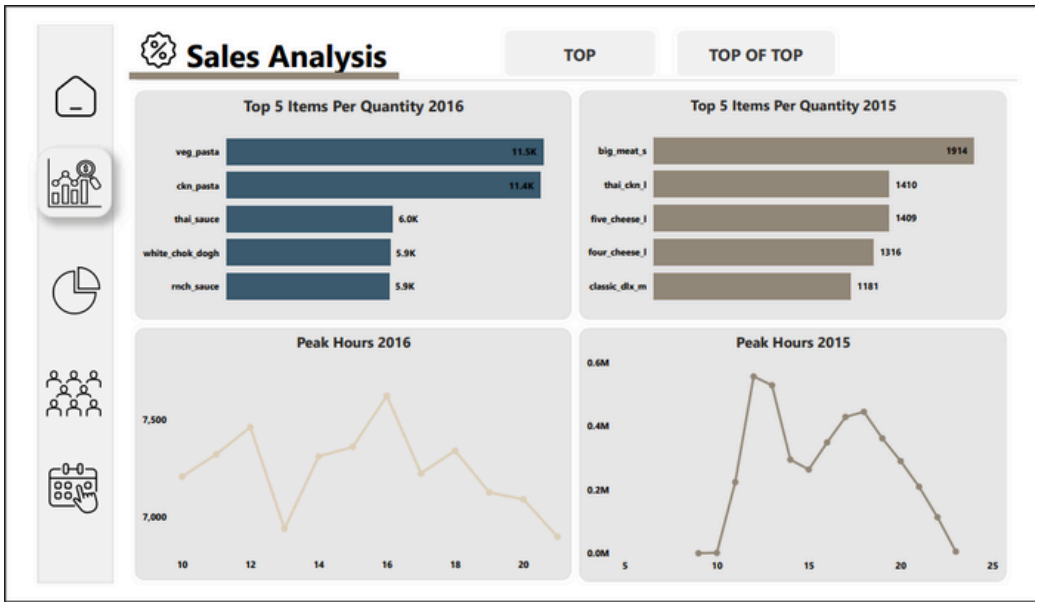
Query Result

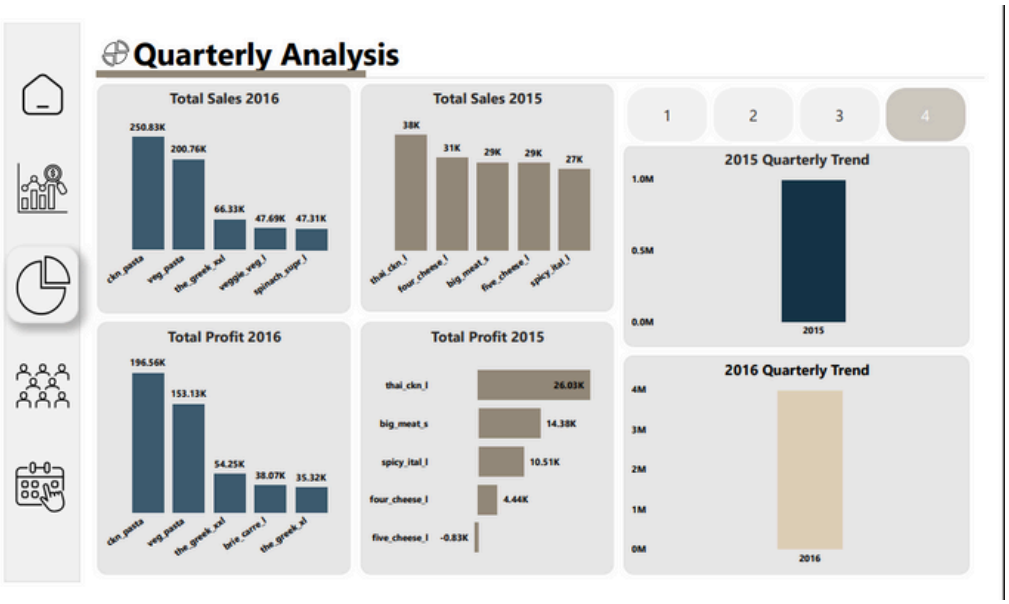
SQL | All Rows Fetched: 6 in 0.337 seconds

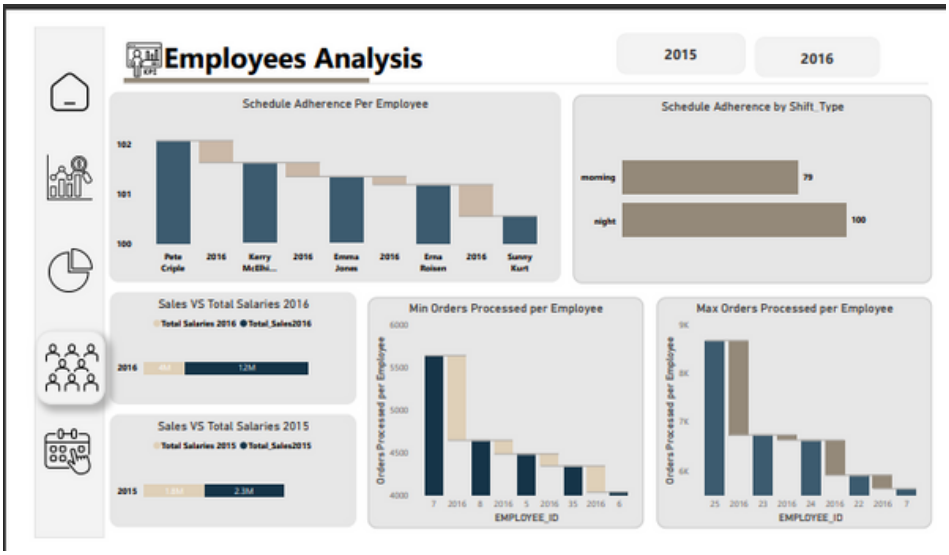
	ORDER_YEAR	ITEM1	ITEM2	PAIR_COUNT
1	2015	biq meat s	thai ckn l	144
2	2015	biq meat s	four cheese l	126
3	2015	biq meat s	five cheese l	125
4	2016	ckn pasta	veq pasta	674
5	2016	veq pasta	white chok doqh	384
6	2016	tri chok doqh	veq pasta	370

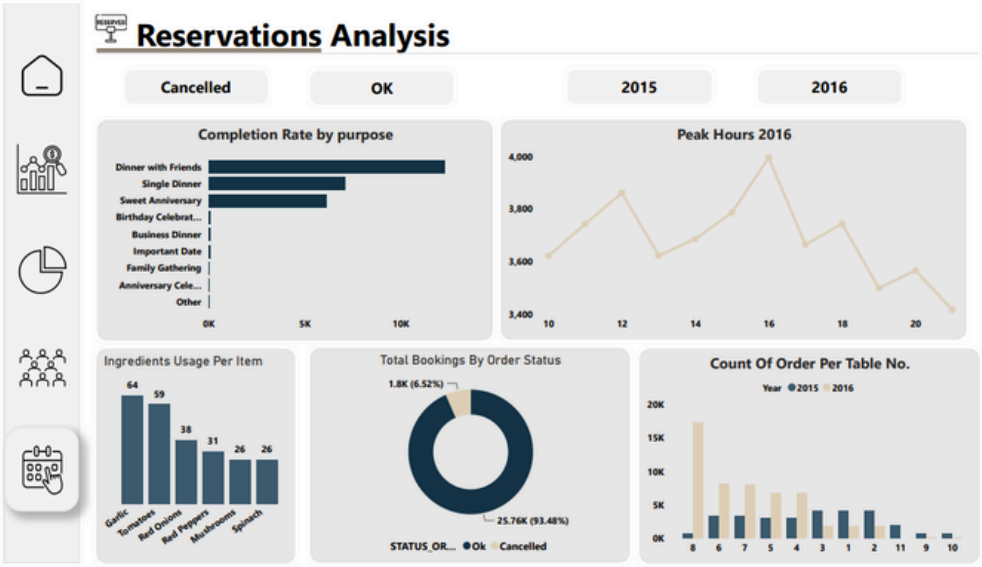
5.1. POWER BI Dashboard





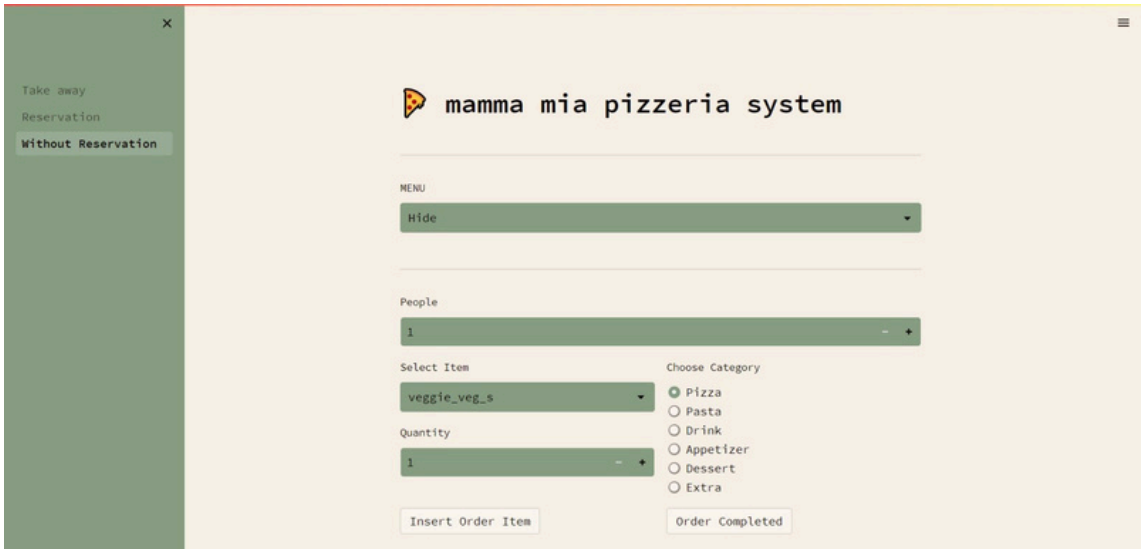


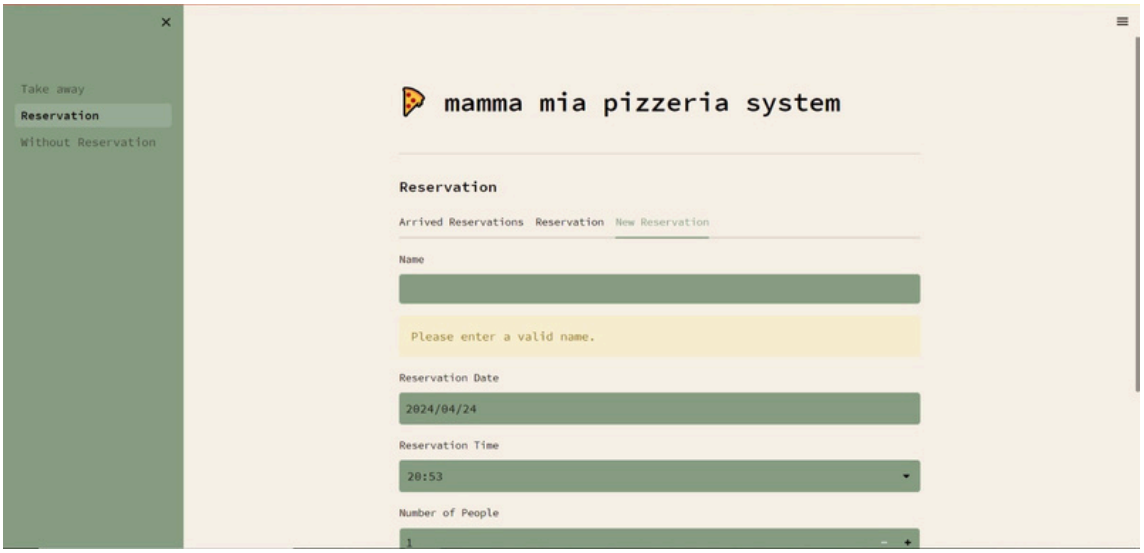


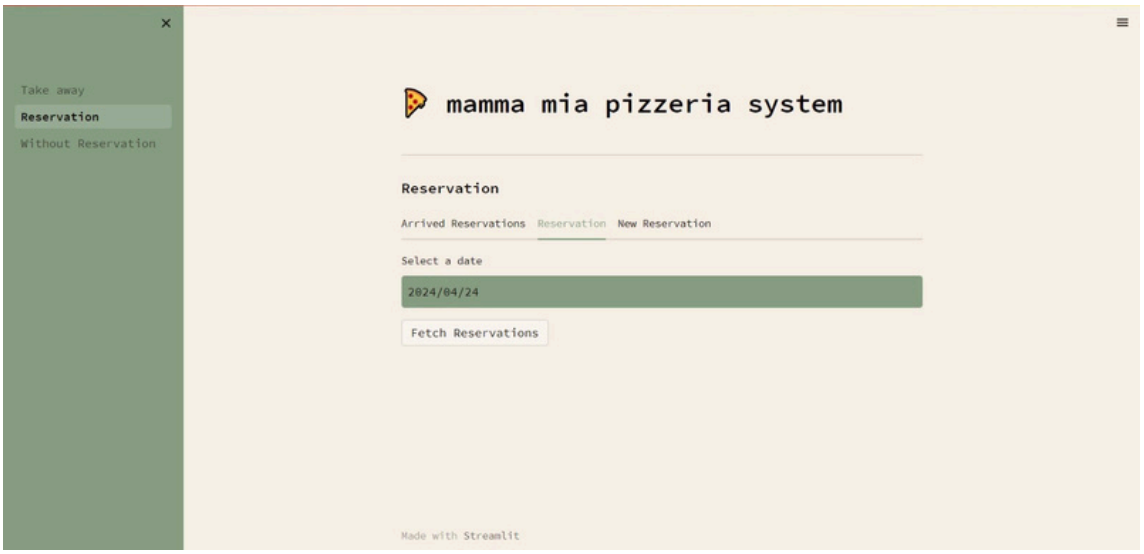


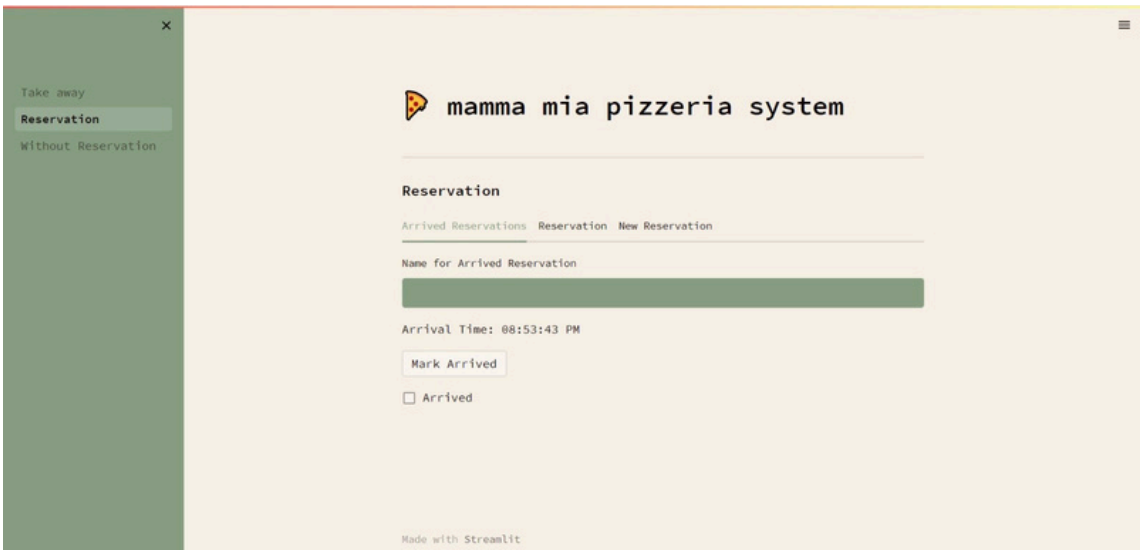
6. Live Application

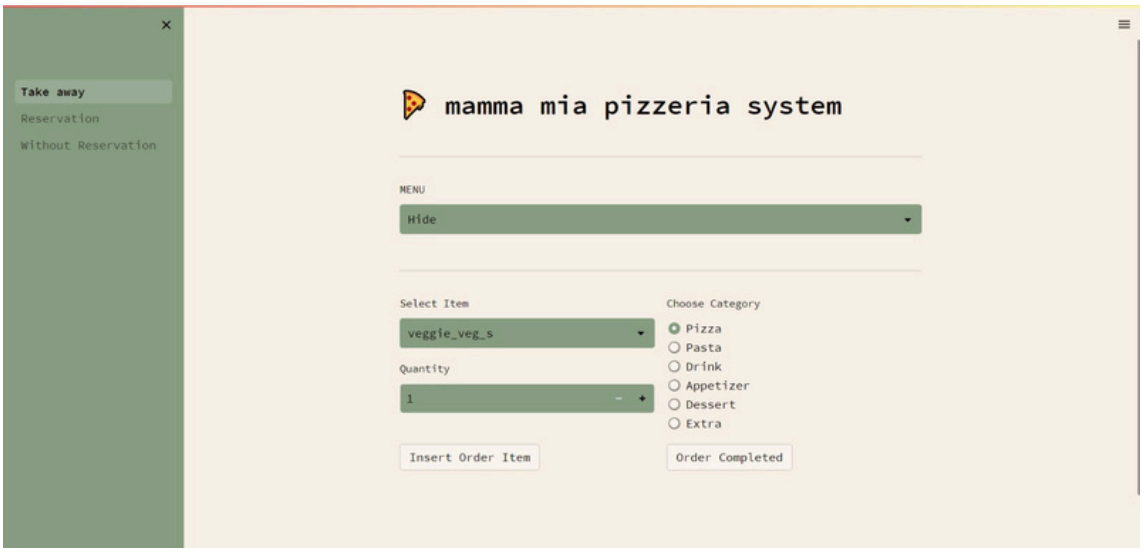
6.1. Demo:
A cashier system that's integrated with the used database to handle Orders Management, Reservations, Kitchen Viewing Orders using Python

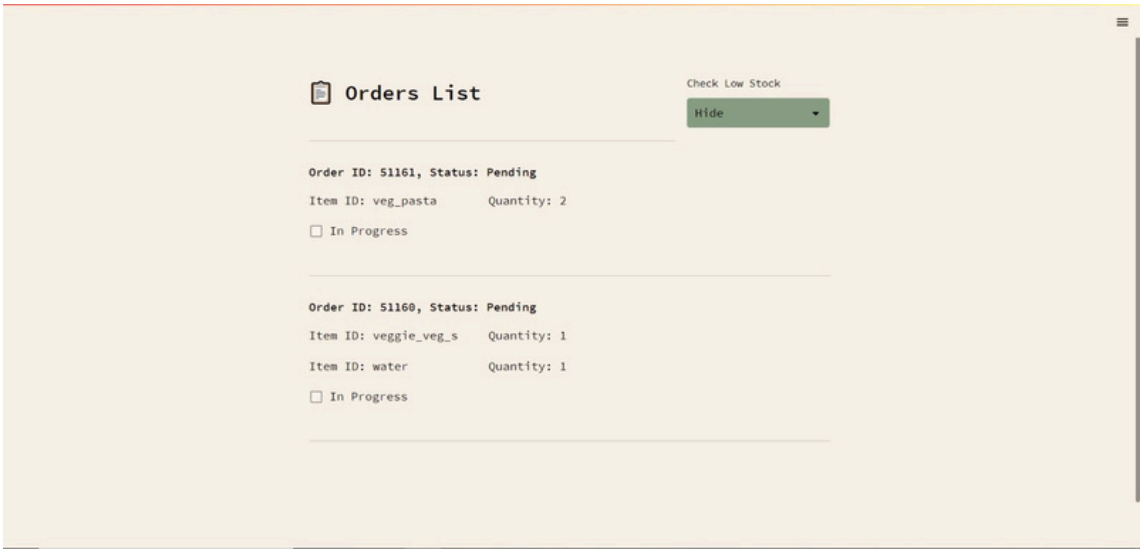



Take away
Reservation
Without Reservation
mamma mia pizzeria system
Reservation
Arrived Reservations
Reservation
New Reservation
Name
Please enter a valid name.
Reservation Date
2024/04/24
Reservation Time
20:53
Number of People
1


Take away
Reservation
Without Reservation
mamma mia pizzeria system
Reservation
Arrived Reservations
Reservation
New Reservation
Select a date
2024/04/24
Fetch Reservations
Made with Streamlit


Take away
Reservation
Without Reservation
mamma mia pizzeria system
Reservation
Arrived Reservations
Reservation
New Reservation
Name for Arrived Reservation
Arrival Time: 08:53:43 PM
Mark Arrived
Arrived
Made with Streamlit





7. Summary

The Restaurant Management System (RMS) project aims to develop a comprehensive solution for restaurant owners and managers to optimize operations, improve efficiency, and enhance customer experience. By addressing challenges such as data scarcity, limited data features, and integration complexities, the project aims to provide robust functionalities for menu management, order processing, reservation handling, inventory control, employee management, analytics, and reporting. Through these efforts, the RMS project seeks to empower restaurants with tools for data-driven decision-making and operational excellence.