USE CASE STUDY REPORT

Group No.: Group 4

Student Names: Shraman Arya – 002101258
Sai Gowtham Babu Amburi - 002190843

Executive Summary:

The primary objective of this study was to design a restaurant database that would help store data of all the orders, food items along with their prices, details of the employees working in that restaurant. This creation of a relational database would help the restaurants maintain a record of all the data at one place.

The database was designed taking the requirements of data fields to handle the restaurant business. The EER and UML diagrams were modelled, followed by the mapping of the conceptual model to a relational model with the required primary and foreign keys. This database was then implemented fully in MySQL environment and a prototype with two tables and one relationship was implemented on MongoDB database to study the feasibility of this database in a NoSQL environment.

The database was created successfully and then connected to R and the analytics was performed to compare various details, some of which have been shown in the study. This database can be further enhanced by analyzing the data using other visualization tools and sales forecasting can be done as to increase the sales of the restaurant and identify the employee's effort.

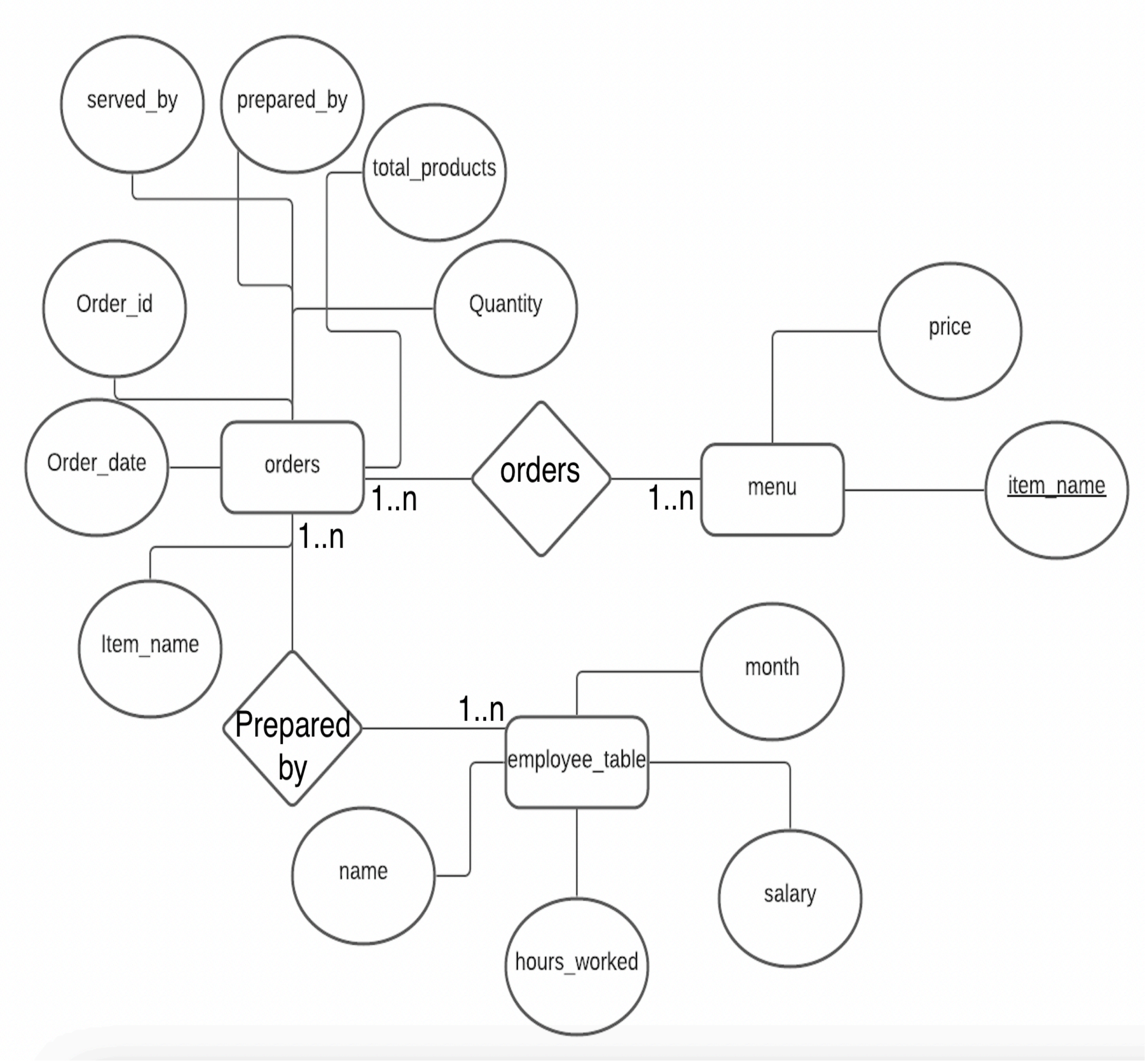
I. Introduction

In the restaurant scenario whenever an order has been placed, the details would be noted and then be stored in a physical storage device. This would not be an ideal case as the data might be lost and may not be recovered. To resolve this issue, we have created a relational database which stores all the data which can be retrieved whenever required without using any physical storage device. This creation of a relational database would resolve the problem of Data redundancy.

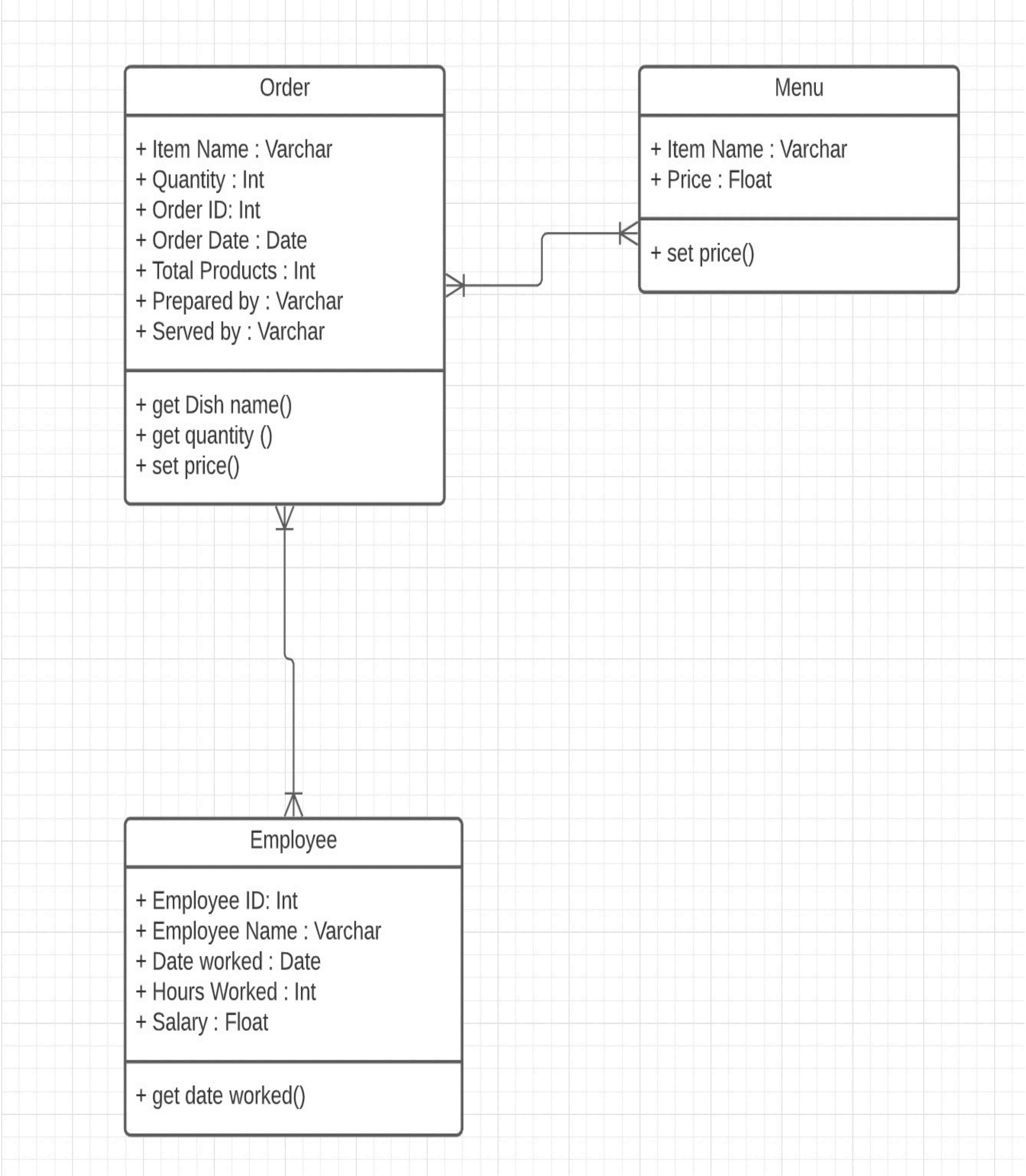
The database was created such that it meets the requirements of the problem statement, we first need to identify the required entities such as order details, price of items, employee details and their work hours. Since, there can be one to many customers ordering food at the same time, there will be a one – many relationships between customers and order. In the same way, two different customers can order same item and a customer can order items multiple times, there will also be a one – many relationships. In order entity, it contains the dish name, quantity and relative price based on dish name and quantity. The total sum of the price will be reflected in customer entity. The price of each item will be taken from price entity where each dish will have a unique ID and relative price associated with it. This also contains a one – many relations as many dishes can be ordered at the same time and vice versa. Coming to the employer information, an entity contains all the details of the employee and maps it to the customer entity. In customer entity, an attribute is created detailing who have made the dish (a dish can be made by one - many chefs). Another entity details the time and date each employee worked.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key- Underlined **Foreign Key- *Italicized***

Order (Order ID , Quantity, *Dish Name*, Total Products, Prepared by, Served by, Order Date)
FOREIGN KEY Dish Name refers to Dish Name in Property; NULL NOT ALLOWED

Price (Price, Dish name)

Employee (Employee ID, Name, Date Worked, Hours Worked, Salary)

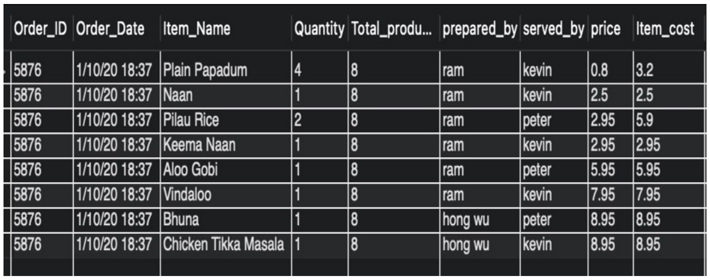
IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

1. Find the order details with order number 5876.

```
SELECT o.*,m.price,
(o.quantity*m.price) as Item_cost
FROM orders o, menu m
where o.item_name=m.item_name
and order_id = 5876;
```

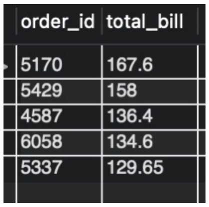


Order_ID	Order_Date	Item_Name	Quantity	Total_produ...	prepared_by	served_by	price	Item_cost
5876	1/10/20 18:37	Plain Papadum	4	8	ram	kevin	0.8	3.2
5876	1/10/20 18:37	Naan	1	8	ram	kevin	2.5	2.5
5876	1/10/20 18:37	Pilau Rice	2	8	ram	peter	2.95	5.9
5876	1/10/20 18:37	Keema Naan	1	8	ram	kevin	2.95	2.95
5876	1/10/20 18:37	Aloo Gobi	1	8	ram	peter	5.95	5.95
5876	1/10/20 18:37	Vindaloo	1	8	ram	kevin	7.95	7.95
5876	1/10/20 18:37	Bhuna	1	8	hong wu	peter	8.95	8.95
5876	1/10/20 18:37	Chicken Tikka Masala	1	8	hong wu	kevin	8.95	8.95

2. Find the order_id with total bill amount in descending order and limit to the top 5 records.

```
with cte as (select  o.order_id, o.item_name, o.quantity, o.total_products, m.price,
                     (o.quantity*m.price) as bill_amount
                     from orders o
                     join menu m
                     on o.Item_Name = m.Item_Name)
select order_id, round(sum(bill_amount), 2) as total_bill
from cte
group by order_id
```

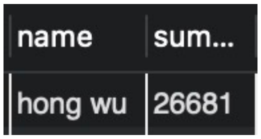
```
order by total_bill desc
limit 5;
```



order_id	total_bill
5170	167.6
5429	158
4587	136.4
6058	134.6
5337	129.65

3. Find the details of an employee with highest salary.

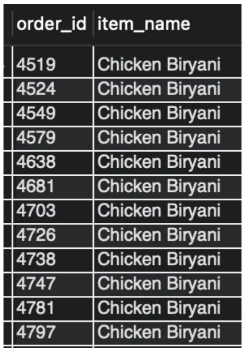
```
select name, sum(salary) from employee_table
group by name
order by sum(salary) desc
LIMIT 1;
```



name	sum...
hong wu	26681

4. Find the orders with item name chicken biryani and having count greater than 5.

```
select order_id, item_name
from orders
where order_id in (select order_id
                              from orders
                              group by order_id
having count(order_id)>5) and item_name = ‘Chicken Biryani’;
```

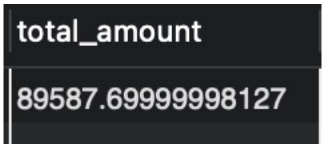


order_id	item_name
4519	Chicken Biryani
4524	Chicken Biryani
4549	Chicken Biryani
4579	Chicken Biryani
4638	Chicken Biryani
4681	Chicken Biryani
4703	Chicken Biryani
4726	Chicken Biryani
4738	Chicken Biryani
4747	Chicken Biryani
4781	Chicken Biryani
4797	Chicken Biryani

5. Find the total amount earned by the restaurant in that particular year.

```
with cte as (select o.order_id,  o.item_name,  o.quantity,  o.total_products, o.order_date, m.price,
              (o.quantity*m.price) as bill_amount
              from orders o
              join menu m
              on o.Item_Name = m.Item_Name)
select sum(bill_amount) as total_amount
```

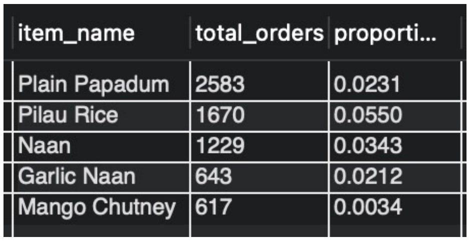
```
from cte ;
```



total_amount
89587.69999998127

6. **Find the 5 most ordered item and what its proportion in the total data base.**

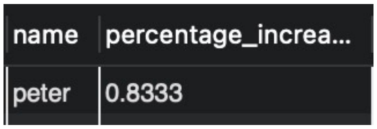
```
 with a as(select o.item_name, sum(o.quantity) as total_orders, sum((o.quantity * m.price)) as total_price
 from orders o
 join menu m
 on o.item_name = m.item_name
 group by item_name
 order by total_orders desc)
select item_name, total_orders, cast((total_price/(select sum(total_price) from a)) as decimal(7,4)) as proportion
from a
limit 5;
```



item_name	total_orders	proporti...
Plain Papadum	2583	0.0231
Pilau Rice	1670	0.0550
Naan	1229	0.0343
Garlic Naan	643	0.0212
Mango Chutney	617	0.0034

7. **Find the employee whose salary increased by the most percentage by start date to end data.**

```
 select name, (min(salary)/ max(salary)) as percentage_increase
 from employee_table
 group by name
 order by percentage_increase desc
 limit 1;
```



name	percentage_increa...
peter	0.8333

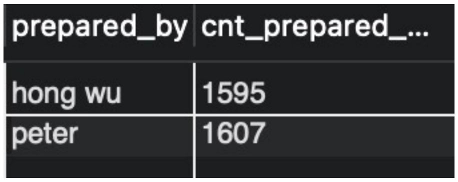
8. **Find the month where the max orders are placed and give the chefs and server name who prepared and served the food**

```
 with c as (with b as (with a as (select order_id, item_name, quantity, prepared_by, served_by,
 case
   when length(order_date) = 16 then right(left(order_date, 5) ,2)
   when length(order_date) = 12 then right(left(order_date, 3) ,2)
   when length(order_date) = 14 then right(left(order_date, 5) ,2)
   when length(order_date) = 13 then right(left(order_date, 4) ,2)
```

```
end as month_number
from orders)
select order_id, item_name, quantity, prepared_by, served_by, replace(month_number, '/',0) as month from a)
```

from the analysis we came to know that December is having more orders with 3986. So, we used month = 12 for analysis purpose.

```
select * from b where month = 12)
(select prepared_by, count(order_id) as cnt_prepared_by
from c
group by prepared_by
order by cnt_prepared_by desc
limit 1) union all (select served_by, count(order_id) as cnt_served_by
from c
group by served_by
order by cnt_served_by desc
limit 1);
```



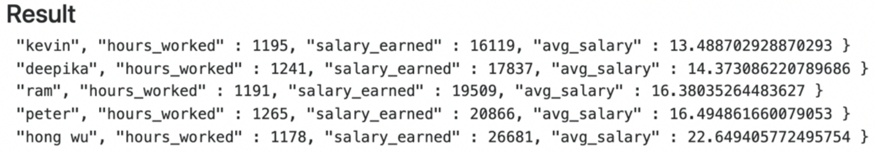
prepared_by	cnt_prepared_...
hong wu	1595
peter	1607

NoSQL Implementation:

We have created our restaurant database in MongoDB and added employee details who have been working in that restaurant. By writing queries in MongoDB, we were able to identify the position of an employee based on his working hours, salary earned and average salary per hour.

1. **Total worked hours, total salary and average salary across all the months.**

```
db.res.aggregate([
   { "$group": {
      "_id":  '$name' ,
      'hours_worked': { $sum: "$hours_worked" },
       'salary_earned': { $sum: "$salary" }
   } },
   { "$addFields": {
      "avg_salary": { "$divide": ["$salary_earned", '$hours_worked'] },
   } },
    {$sort:{avg_salary:1}}
])
```

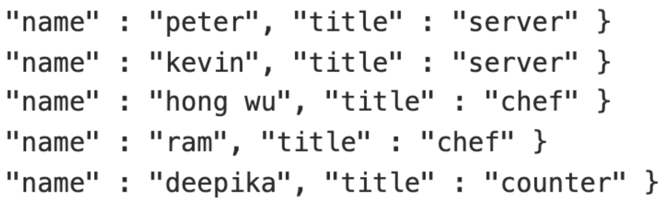


Result

```
"kevin", "hours_worked" : 1195, "salary_earned" : 16119, "avg_salary" : 13.488702928870293 }
"deepika", "hours_worked" : 1241, "salary_earned" : 17837, "avg_salary" : 14.373086220789686 }
"ram", "hours_worked" : 1191, "salary_earned" : 19509, "avg_salary" : 16.38035264483627 }
"peter", "hours_worked" : 1265, "salary_earned" : 20866, "avg_salary" : 16.494861660079053 }
"hong wu", "hours_worked" : 1178, "salary_earned" : 26681, "avg_salary" : 22.649405772495754 }
```

2. **The reason for higher salary for hong wu?**

db.details_employee.find();



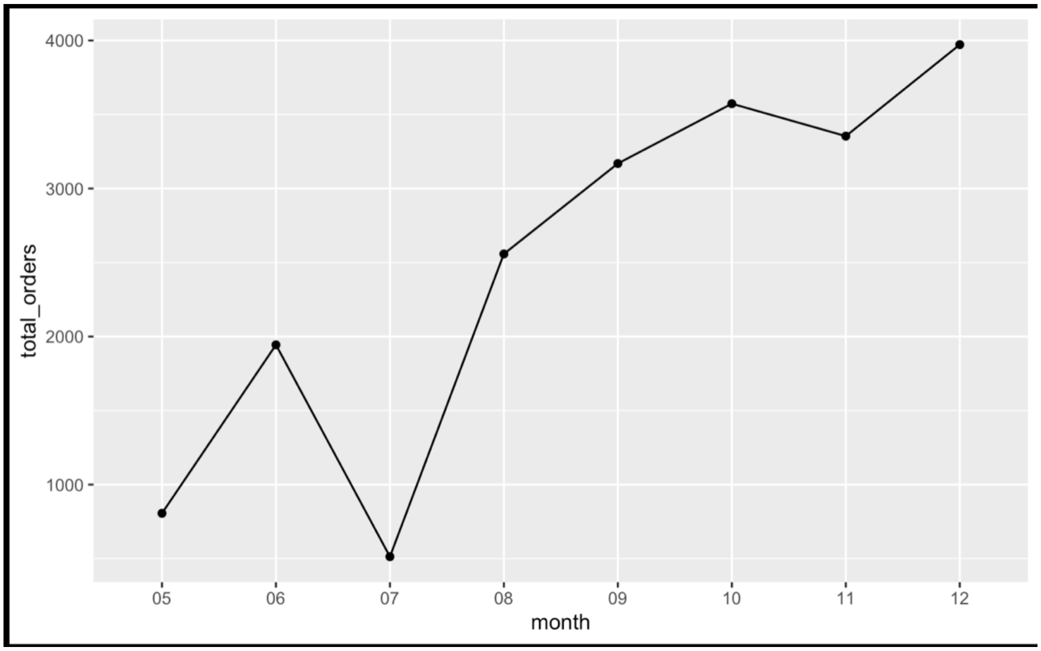
```
"name" : "peter", "title" : "server" }
"name" : "kevin", "title" : "server" }
"name" : "hong wu", "title" : "chef" }
"name" : "ram", "title" : "chef" }
"name" : "deepika", "title" : "counter" }
```

from the analysis it can be seen that hong wu is the chef and thus is getting paid more compared to all other workers.

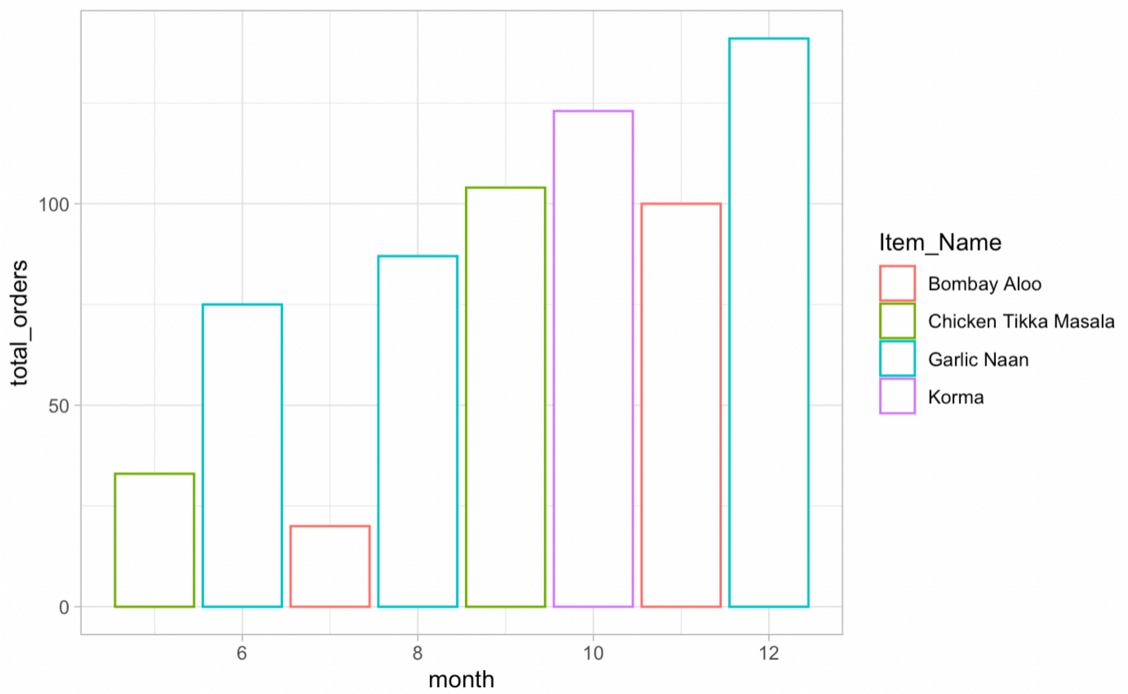
V. Database Access via R

The database is accessed using r and visualization of analyzed data is shown below. The connection of MySQL to R is done using spark_connect and the data base if loaded into the spark using spark_read_csv.

1. **Sum of total orders across different months.**



2. Most ordered item with respect to each month having total orders less than 1000.



VII. Summary and Recommendation

The International Tourist database designed on MySQL is an industry ready relational database that can be implemented in the Indian tourism industry. It will result in great cost savings in the tourist data input process and provides great analytics capabilities, a small part of which is shown in this report utilizing Python and Tableau. We went an extra step to design the front end UX wireframes for the database as well using Balsamiq Wireframe software.

Improvement on the database would be the implementation of data governance measures on the database to ensure data quality as the entered tourist data is reused by other properties to enter the stay details into the database of that tourist. The Machine-Readable Zone of Passport using Python's Passport-Eye package is an option to exercise a data governance to ensure data quality of the primary key and attributes of the tourist data like Name, Date of Birth and Passport Expiry Date.

The shortcoming would in the NoSQL implementation of this Database on Neo4j. More study should be done on how a unique relational database like this can be implemented in a NoSQL environment, especially the 'stay' table as that refers to data from both 'property' and 'international tourist' tables as foreign keys, along with stay dates updated in real-time as attributes within the stay relationship. Having implemented a few tables on Neo4j graph database and using Cypher to query on it, this can certainly be used to build the database but the whole data input process benefits of using a relational database and (n-n) relationships that are constantly updated in the stay data table has greater advantages in the real world application, and this outweighs the benefits from a NoSQL database for the same.