USE CASE STUDY REPORT

Group 22

Paka Vishesh & Jahnavi Chowdary T

I. INTRODUCTION (BUSINESS PROBLEM DEFINITION)

Today, managing a supermarket is quite challenging due to the need to monitor numerous databases. To manage and operate the store profitably, the manager must keep track on many factors. One must track the availability of different products in different categories and should have a record of the expiry dates of each product. The amount of additional inventory to order or produce, as well as when to do so, are also things he must know.

Businesses usually keep track of the products supplied on paper invoices, which are easily lost and may lead to understocking or overstocking of the products. So, in order to increase the functionality of store administration, we recommend our project, which aids owners in keeping track of all necessary data. Thus, the store management system in the problem statement is store management software that aids in the examination of numerous elements that can increase store productivity and sales. A business strategy that will increase the store's sales can be developed using the data from the databases.

The primary objective of this project is to design application and implement a relational database that will help the store managers and owners to track different works and transactions of different stores at a time. Designing such application helps them to have a better view of their business. A departmental store's general operation and function rely primarily on the administration and management of the store. There will be many products and categories, especially in the case of larger stores. Therefore, efficient management and good resource management are essential for success and smooth operation.

II. CONCEPTUAL DATA MODELING

An EER diagram has been designed to capture all the information required to create the database for Store Management System.

Description:

Every store has unique id (store_id) and a manager to manage the store. Each has a minimum of one employee and every employee has a employee_id to uniquely identify and works at most in one store. There are group of suppliers who are identified by supplier_id and supplies from a range of zero to many stores and every store has more than one supplier. Every supplier supply product which are uniquely identified by product_id and every product can be supplied by only one supplier. Each product belong to a category which is identified by category_id and a category can have multiple products. Every customer is identified by customer_id and enrolls in to a plan which identified by plan_id and a plan can hold multiple customers. Every plan has different number of promotions based on the subscription amount and term of plan and a particular promotion can be in more than one plan.

Every Customer can choose a product from a range of zero to many and add them to a order. Every order is identified by order_id and maintains the product_id and quantity of the product. The discount percentage from promotion is applied on order total and an invoice is generated which is identified by invoice_id and also maintains employee who billed the invoice for any future analytical purposes.

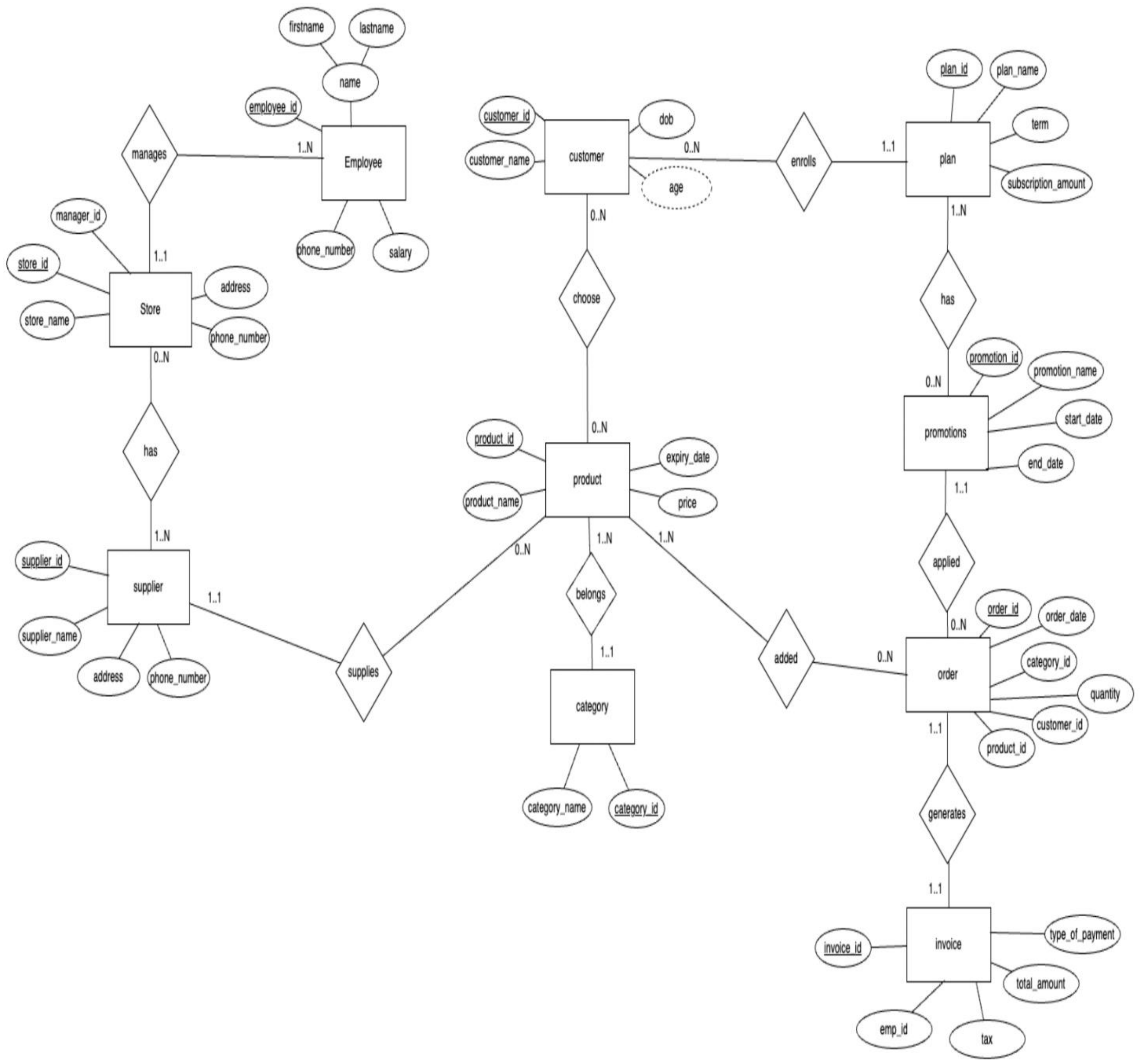


Figure 1: *EER*

UML DIAGRAM

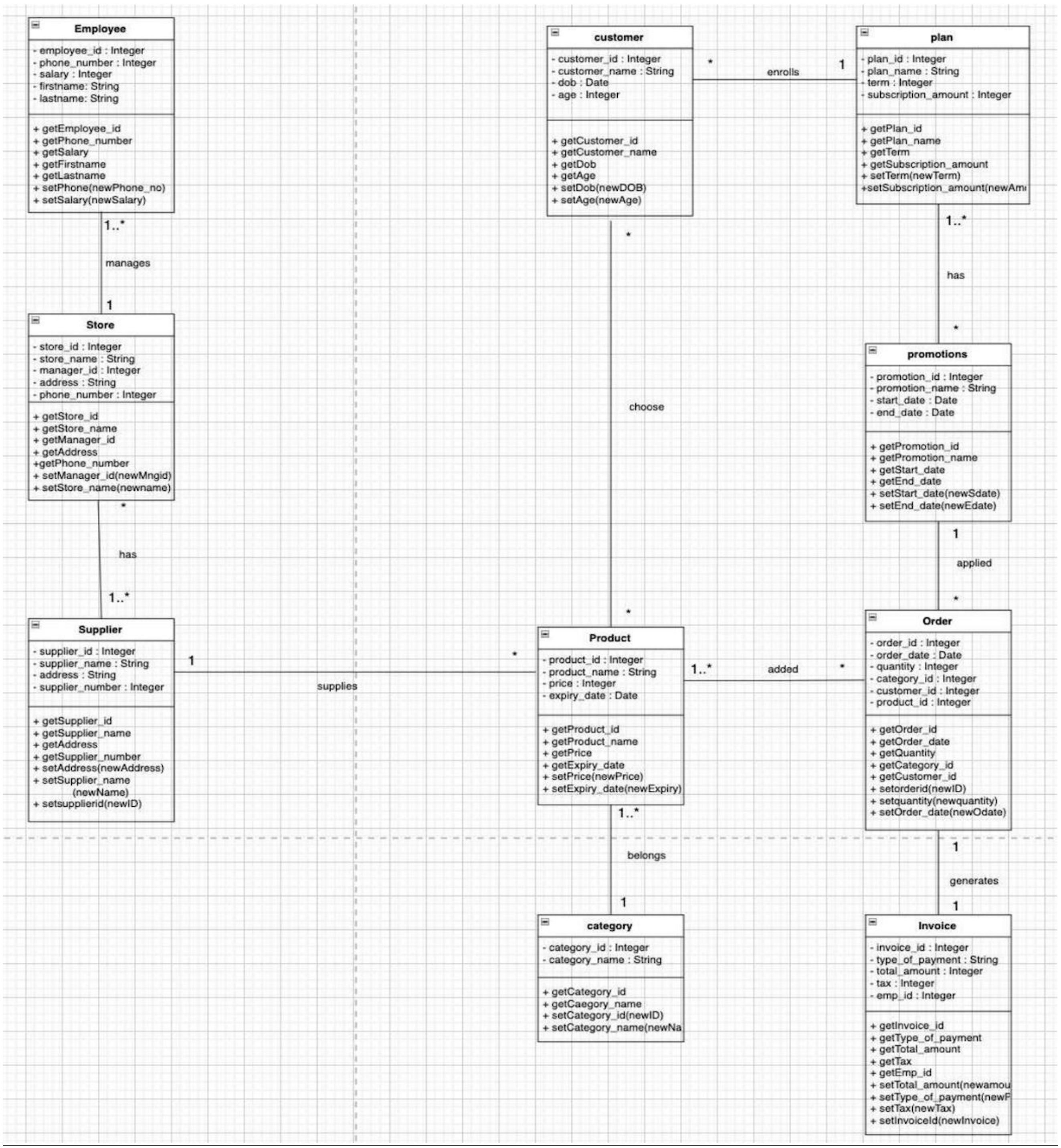


Figure 2: *UML*

III. MAPPING CONCEPTUAL MODEL TO RELATIONAL MODEL

Note: Primary keys are underlined and foreign keys are written in red color

EMPLOYEE (Employee_id, firstname, lastname, salary,phone_number, store_id)

- Employee_id is the primary key.
- Store_id foreign key references STORE (store_id)

STORE (store_id, store_name, manager_id, phone_number, address)

- Store_id is the primary key.
- manager_id foreign key references EMPLOYEE (Employee_id)

SUPPLIER (supplier_id, supplier_name, address, phone_number)

- supplier_id is the primary key

STORE_SUPPLIER (store_id, supplier_id)

- store_id foreign key references STORE (store_id)
- supplier_id foreign key references SUPPLIER (supplier_id)
- Together store_id and supplier_id is the primary key

PRODUCT (product_id, product_name, price, expiry_date,category_id, supplier_id)

- Product_id is the primary key
- Category_id foreign key references CATEGORY (category_id)
- Supplier_id foreign key references SUPPLIER (supplier_id)

CATEGORY (category_id, category_name)

- Category_id is the primary key

CUSTOMER (customer_id, customer_name, date_of_birth, age,plan_id)

- Customer_id is the primary key
- Plan_id foreign key references PLAN (plan_id)

CUSTOMER_PRODUCT (customer_id, product_id)

- Customer_id foreign key references CUSTOMER (customer_id)
- Product_id foreign key references PRODUCT (product_id)
- Together customer_id, product_id is the primary key

PLAN (plan_id, plan_name, term, subscription_amount)

- Plan_id is the primary key

PROMOTIONS (promotion_id, promotion_name, start_date,end_date, discount_percentage)

- Promotion_id is the primary key

PLAN_PROMOTION (plan_id, promotion_id)

- Plan_id foreign key references PLAN (plan_id)
- Promotion_id foreign key references PROMOTION(promotion_id)
- Together plan_id and promotion_id is the primary key

ORDER (order_id, order_date, customer_id, promotion_id)

- Order_id is the primary key
- Customer_id foreign key references CUSTOMER (customer_id)
- Promotion_id foreign key references PROMOTION(promotion_id)

ORDER_ITEMS (order_id, category_id, product_id, quantity)

- Order_id foreign key references ORDER (order_id)
- Product_id foreign key references PRODUCT (product_id)
- Category_id foreign key references CATEGORY (category_id)

- Together order_id, product_id, category_id is the primary key

INVOICE (invoice_id, employee_id, order_id, total_amount, tax,type_of_payement)

- Invoice_id is the primary key
- Employee_id foreign key references EMPLOYEE (employee_id)
- Order_id foreign key references ORDER (order_id)

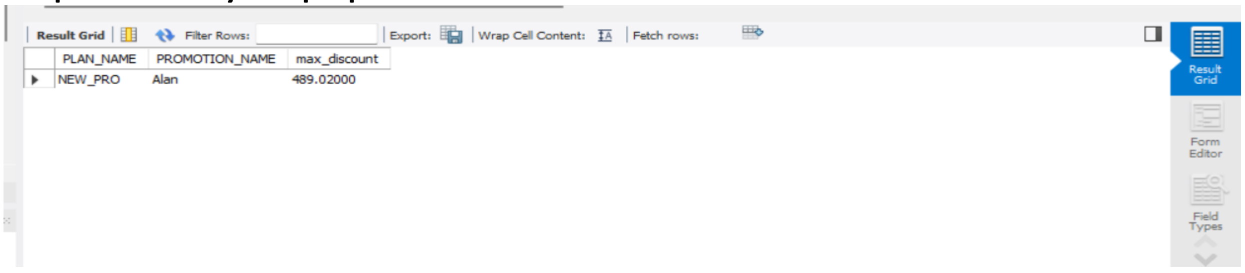
IV. IMPLEMENTATION OF RELATIONAL MODEL VIA MYSQL ANDNOSQL

Implementation in MYSQL

QUERY 1:

```
SELECT P.PLAN_NAME, PR.PROMOTION_NAME,(I.TOTAL_AMOUNT * 0.01 *
PR.DISCOUNT_PERCENTAGE) AS MAX_DISCOUNT FROM INVOICE I, ORDERS
O,CUSTOMER C  ,PLAN P, PROMOTIONS PR,PLAN_PROMOTIONS PP WHERE
I.ORDER_ID=O.ORDER_ID AND O.CUSTOMER_ID = C.CUSTOMER_ID AND C.PLAN_ID =
P.PLAN_ID AND P.PLAN_ID = PP.PLAN_ID  AND PP.PROMOTION_ID =
PR.PROMOTION_ID ORDER BY MAX_DISCOUNT DESC LIMIT 1;
```

Output and Analytical purpose:



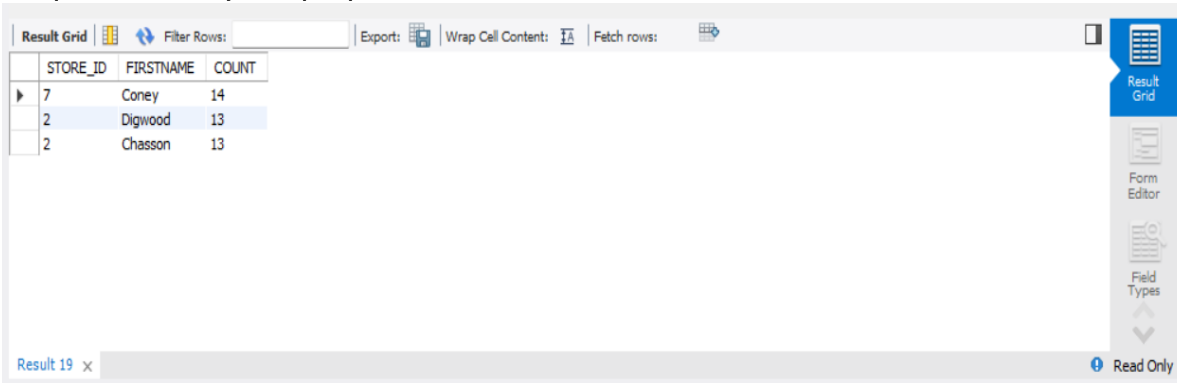
PLAN_NAME	PROMOTION_NAME	max_discount
NEW_PRO	Alan	489.02000

The analytical purpose of this query is to find the maximum discount given to a customer with this we can analyze from which plan and promotion he got benefited and we can advertise this plan saying maximum savings to attract customers and also improve other plans to attract more customers.

QUERY2:

```
SELECT E.STORE_ID,E.FIRSTNAME, COUNT(I.EMPLOYEE_ID) COUNT FROM EMPLOYEE
E, INVOICE I WHERE I.EMPLOYEE_ID=E.EMPLOYEE_ID GROUP BY I.EMPLOYEE_ID
ORDER BY COUNT DESC LIMIT 3;
```

Output and Analytical purpose:



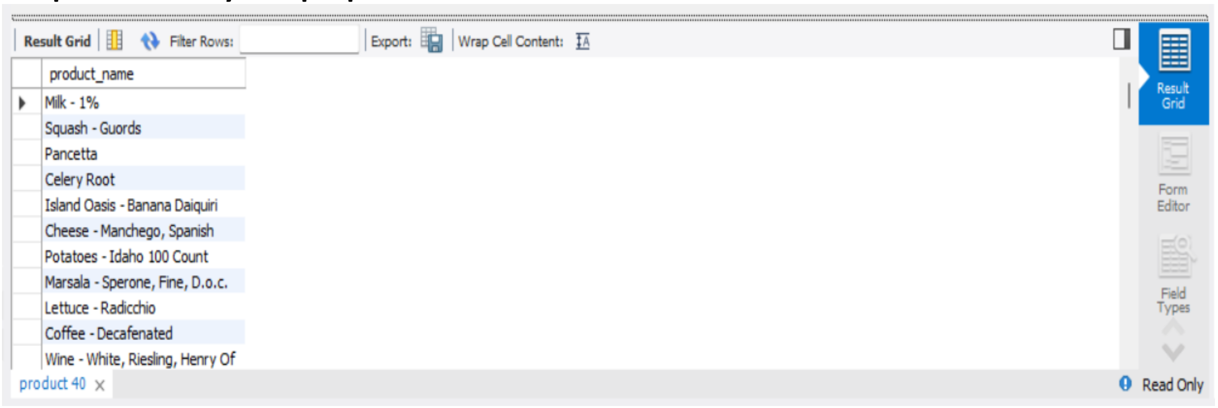
STORE_ID	FIRSTNAME	COUNT
7	Coney	14
2	Digwood	13
2	Chasson	13

The Analytical purpose of this query is to find the employees who had billed most invoices with which we can identify that employee as employee of the month and give a bonus which will bring a competitive environment among the employees.

QUERY3:

SELECT P.PRODUCT_NAME FROM PRODUCT P WHERE (SELECT COUNT(*) FROM ORDER_ITEMS OI WHERE P.PRODUCT_ID=OI.PRODUCT_ID GROUP BY OI.PRODUCT_ID)>=20;

Output and Analytical purpose:



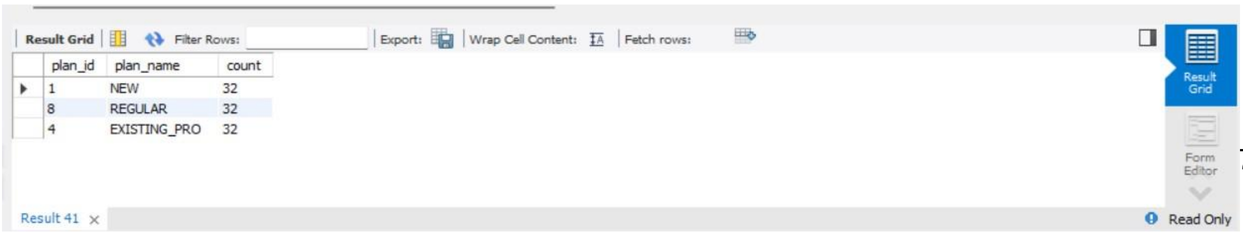
product_name
Milk - 1%
Squash - Guords
Pancetta
Celery Root
Island Oasis - Banana Daiquiri
Cheese - Manchego, Spanish
Potatoes - Idaho 100 Count
Marsala - Sperone, Fine, D.o.c.
Lettuce - Radicchio
Coffee - Decafenated
Wine - White, Riesling, Henry Of

The Analytical purpose of this query is to find the products having at least 20 orders which depicts them as the popular products so that we need to regularly check them and keep in stock.

QUERY4:

SELECT C.PLAN_ID, P.PLAN_NAME, COUNT(C.PLAN_ID) COUNT FROM CUSTOMER C, PLAN P WHERE C.PLAN_ID=P.PLAN_ID GROUP BY C.PLAN_ID ORDER BY COUNT DESC LIMIT 3;

Output and Analytical purpose:

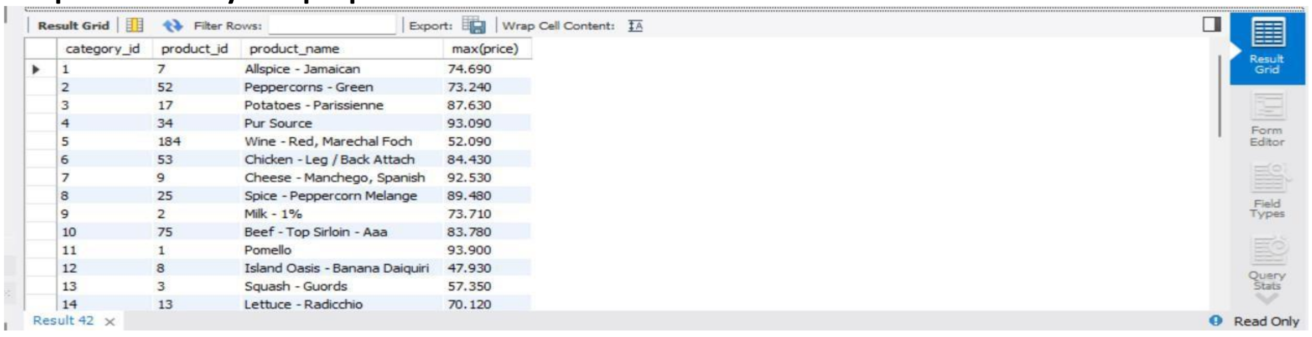


plan_id	plan_name	count
1	NEW	32
8	REGULAR	32
4	EXISTING_PRO	32

The analytical purpose of this query is to find the popular plans among the customers using this we can recommend this plan to new customers or add some more promotions to other plans to attract more customers.

QUERY5:
SELECT CATEGORY_ID, PRODUCT_ID, PRODUCT_NAME, MAX(PRICE) FROM PRODUCT GROUP BY CATEGORY_ID;

Output and Analytical purpose:

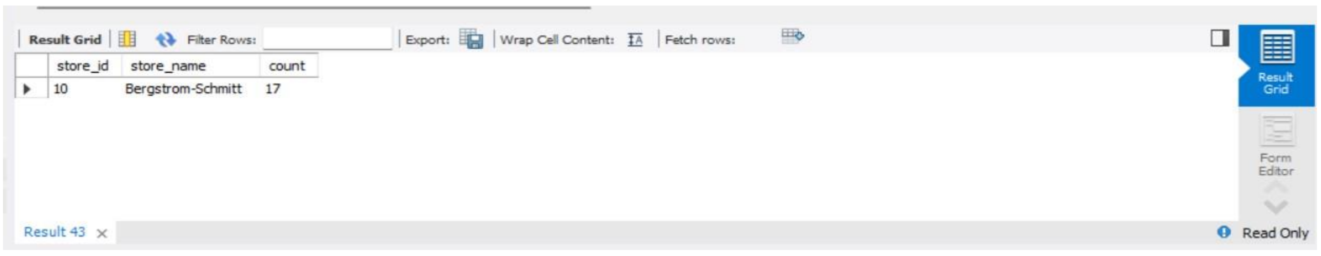


category_id	product_id	product_name	max(price)
1	7	Allspice - Jamaican	74.690
2	52	Peppercorns - Green	73.240
3	17	Potatoes - Parissienne	87.630
4	34	Pur Source	93.090
5	184	Wine - Red, Marechal Foch	52.090
6	53	Chicken - Leg / Back Attach	84.430
7	9	Cheese - Manchego, Spanish	92.530
8	25	Spice - Peppercorn Melange	89.480
9	2	Milk - 1%	73.710
10	75	Beef - Top Sirloin - Aaa	83.780
11	1	Pomello	93.900
12	8	Island Oasis - Banana Daiquiri	47.930
13	3	Squash - Guords	57.350
14	13	Lettuce - Radicchio	70.120

The analytical purpose of this query is to find the categories with maximum price of products which helps us to give some special offers by merging different categories so that customers does not feel too pricy for single product.

QUERY6:
SELECT E.STORE_ID, S.STORE_NAME,COUNT(E.STORE_ID) COUNT FROM STORE S, EMPLOYEE E WHERE S.STORE_ID=E.STORE_ID GROUP BY E.STORE_ID ORDER BY COUNT DESC LIMIT 1;

Output and Analytical purpose:
The analytical purpose of this query is to find the store with more number of

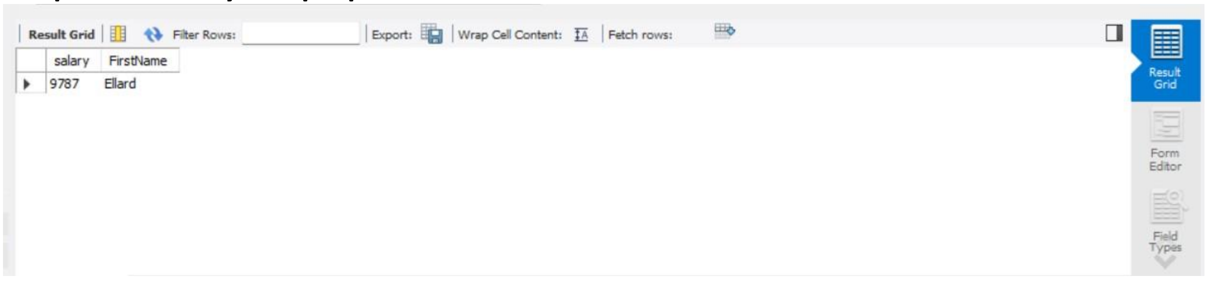


store_id	store_name	count
10	Bergstrom-Schmitt	17

employees depicts that the store is running well and busy or in case of layoffs we can see for a store to first sack the employees.

QUERY7:
SELECT SALARY, FIRSTNAME FROM EMPLOYEE ORDER BY SALARY DESC LIMIT 3,1;

Output and Analytical purpose:



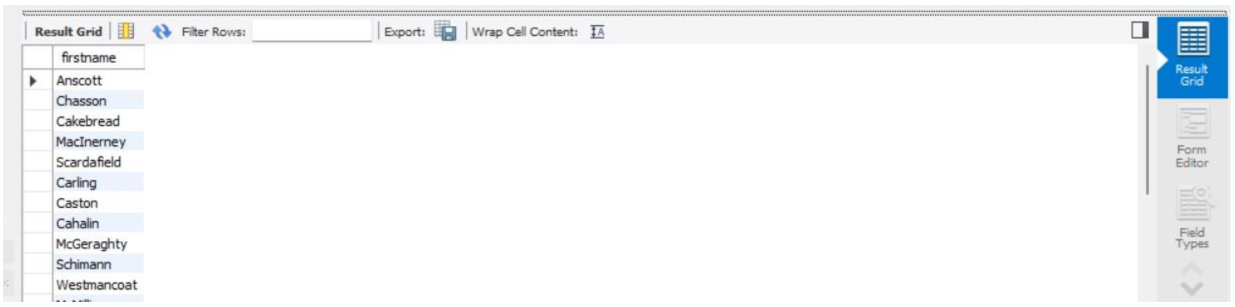
salary	FirstName
9787	Ellard

The analytical purpose of this query is to find the nth highest salary of an employee which helps us to compare salaries between different employees and consider them for managers in next cycle.

QUERY8:

SELECT FIRSTNAME FROM EMPLOYEE WHERE FIRSTNAME LIKE '%A%C%' OR FIRSTNAME LIKE '%C%A%';

Output and Analytical purpose:



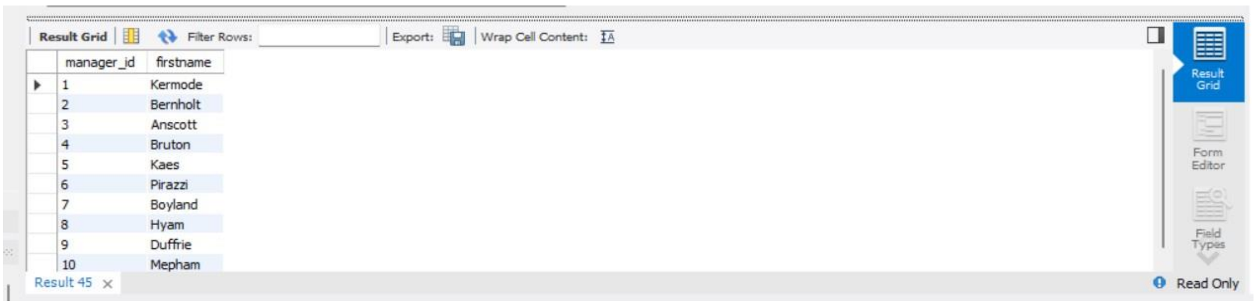
firstname
Anscott
Chasson
Cakebread
MacInerney
Scardafield
Carling
Caston
Cahalin
McGeraghty
Schimann
Westmancoat

The analytical purpose of this query is to filter the employee names on different patterns of names.

QUERY9:

SELECT S.MANAGER_ID, E.FIRSTNAME FROM EMPLOYEE E JOIN STORE S ON E.EMPLOYEE_ID=S.MANAGER_ID;

Output and Analytical purpose:

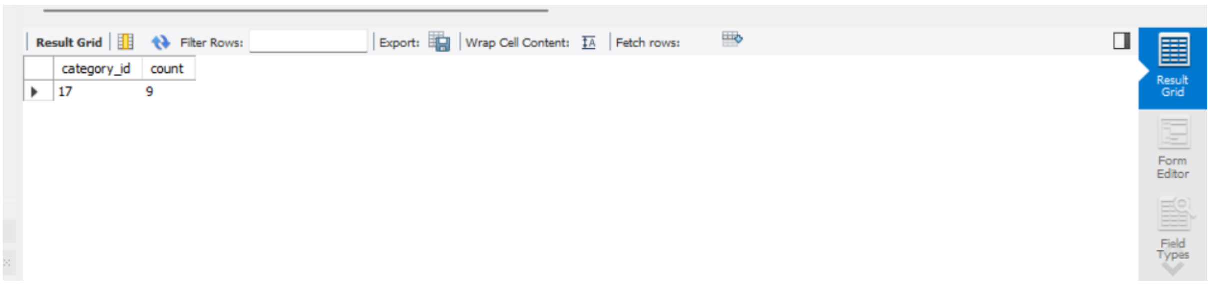


manager_id	firstname
1	Kermode
2	Bernholt
3	Anscott
4	Bruton
5	Kaes
6	Pirazzi
7	Boyland
8	Hyam
9	Duffrie
10	Mepham

The analytical purpose of this query is to find the manager names of all the stores who are also employees in those stores and it is helpful to access them easily.

QUERY10:
SELECT CATEGORY_ID, COUNT(CATEGORY_ID) COUNT FROM PRODUCT GROUP BY CATEGORY_ID ORDER BY COUNT DESC LIMIT 1;

Output and Analytical purpose:



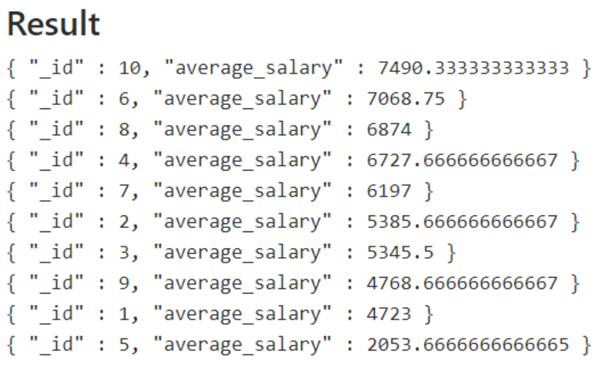
category_id	count
17	9

The analytical purpose of this query is to find the category which has more number of products compared to other category which makes sense to give more space and attention to such large categories to arrange all products in it together.

NOSQL IMPLEMENTATION

QUERY 1:
USING AGGREGATE:
db.employees1.aggregate({"$group": { _id: "$Store_ID", average_salary: { $avg: "$Salary"} } },
{$sort: { average_salary: -1}
});

OUTPUT:

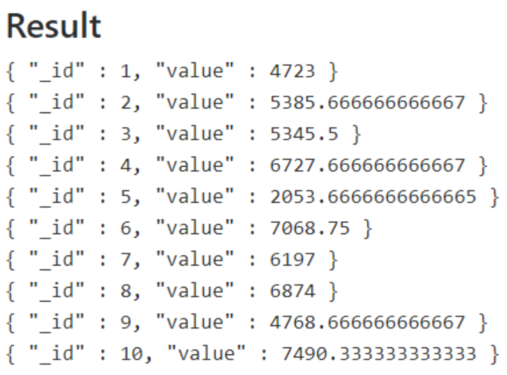


```
Result
{ "_id" : 10, "average_salary" : 7490.333333333333 }
{ "_id" : 6, "average_salary" : 7068.75 }
{ "_id" : 8, "average_salary" : 6874 }
{ "_id" : 4, "average_salary" : 6727.666666666667 }
{ "_id" : 7, "average_salary" : 6197 }
{ "_id" : 2, "average_salary" : 5385.666666666667 }
{ "_id" : 3, "average_salary" : 5345.5 }
{ "_id" : 9, "average_salary" : 4768.666666666667 }
{ "_id" : 1, "average_salary" : 4723 }
{ "_id" : 5, "average_salary" : 2053.6666666666665 }
```

OR

USING MAPREDUCE:
var map=function(){emit(this.Store_ID,this.Salary)};
var reduce=function(Store_ID,Salary){return Array.avg(Salary);};
db.employees1.mapReduce(

```
 map,
 reduce,
 {out:"average_salary"}
 );
db.average_salary.find();
```



Result

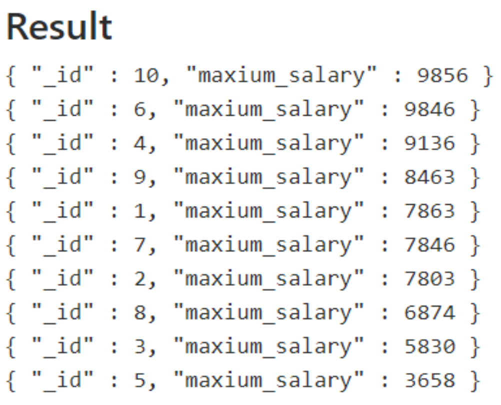
```
{ "_id" : 1, "value" : 4723 }
{ "_id" : 2, "value" : 5385.666666666667 }
{ "_id" : 3, "value" : 5345.5 }
{ "_id" : 4, "value" : 6727.666666666667 }
{ "_id" : 5, "value" : 2053.6666666666665 }
{ "_id" : 6, "value" : 7068.75 }
{ "_id" : 7, "value" : 6197 }
{ "_id" : 8, "value" : 6874 }
{ "_id" : 9, "value" : 4768.666666666667 }
{ "_id" : 10, "value" : 7490.333333333333 }
```

ANALYTICAL PURPOSE:
The Analytical Purpose of this query is to find the average salary of employees according to the store and from this we can depict that store 10 has high average salary compared to other stores which implies the store is running very profitably and employees are experienced.

QUERY2:

```
db.employees1.aggregate ( { "$group": { _id: "$Store_ID", maxium_salary: { $max:
"$Salary"} } },
{$sort: {maxium_salary: -1}
});
```

OUTPUT:



Result

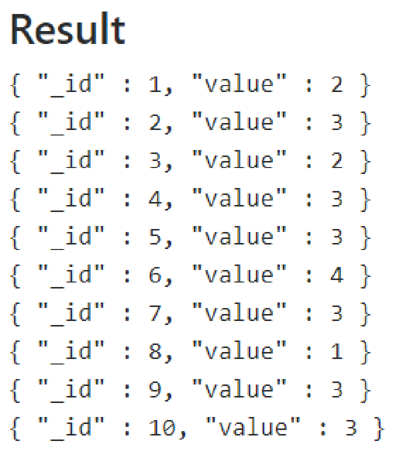
```
{ "_id" : 10, "maxium_salary" : 9856 }
{ "_id" : 6, "maxium_salary" : 9846 }
{ "_id" : 4, "maxium_salary" : 9136 }
{ "_id" : 9, "maxium_salary" : 8463 }
{ "_id" : 1, "maxium_salary" : 7863 }
{ "_id" : 7, "maxium_salary" : 7846 }
{ "_id" : 2, "maxium_salary" : 7803 }
{ "_id" : 8, "maxium_salary" : 6874 }
{ "_id" : 3, "maxium_salary" : 5830 }
{ "_id" : 5, "maxium_salary" : 3658 }
```

ANALYTICAL PURPOSE:
The Analytical purpose of this query is to find the employee having maximum salary at each store. With this we get to know more about that employee and in tough times to cut the management cost also this analytical purpose is useful.

QUERY3:

```
var map = function() {emit(this.Store_ID, 1)};
var reduce = function(key, values) {
 var count = 0;
 values.forEach(function(i) {
count =count+i; });
return count; };
db.employees1.mapReduce(map, reduce, {out: "count_of_employees"}).find();
```

OUTPUT:



Result

```
{ "_id" : 1, "value" : 2 }
{ "_id" : 2, "value" : 3 }
{ "_id" : 3, "value" : 2 }
{ "_id" : 4, "value" : 3 }
{ "_id" : 5, "value" : 3 }
{ "_id" : 6, "value" : 4 }
{ "_id" : 7, "value" : 3 }
{ "_id" : 8, "value" : 1 }
{ "_id" : 9, "value" : 3 }
{ "_id" : 10, "value" : 3 }
```

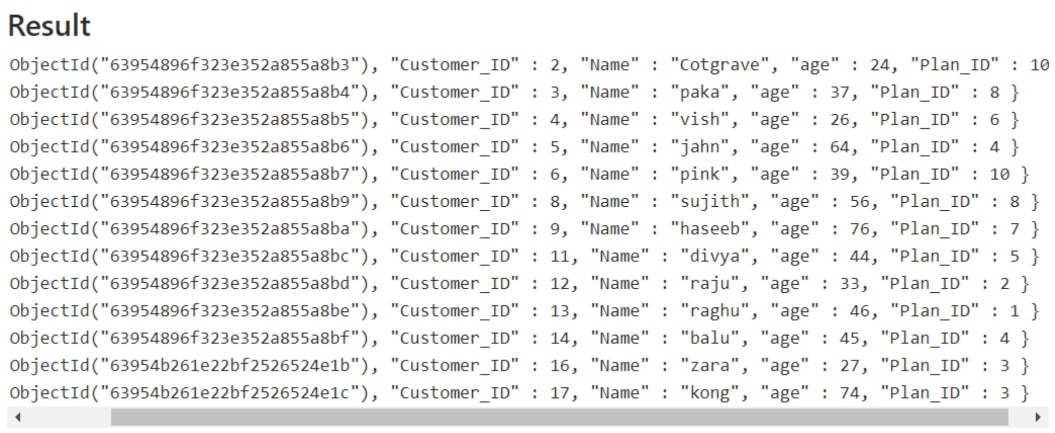
ANALYTICAL PURPOSE:

The analytical purpose of this query is to find the number of employees working in each store which implies how well the store is working and managed, and owner knows the exact amount of employees working and in future if there is any loss then he will have an idea from where to cut the cost.

QUERY4:

```
db.customers.find({age:{$gt: 21}})
```

OUTPUT:



Result

```
ObjectId("63954896f323e352a855a8b3"), "Customer_ID" : 2, "Name" : "Cotgrave", "age" : 24, "Plan_ID" : 10
ObjectId("63954896f323e352a855a8b4"), "Customer_ID" : 3, "Name" : "paka", "age" : 37, "Plan_ID" : 8 }
ObjectId("63954896f323e352a855a8b5"), "Customer_ID" : 4, "Name" : "vish", "age" : 26, "Plan_ID" : 6 }
ObjectId("63954896f323e352a855a8b6"), "Customer_ID" : 5, "Name" : "jahn", "age" : 64, "Plan_ID" : 4 }
ObjectId("63954896f323e352a855a8b7"), "Customer_ID" : 6, "Name" : "pink", "age" : 39, "Plan_ID" : 10 }
ObjectId("63954896f323e352a855a8b9"), "Customer_ID" : 8, "Name" : "sujith", "age" : 56, "Plan_ID" : 8 }
ObjectId("63954896f323e352a855a8ba"), "Customer_ID" : 9, "Name" : "haseeb", "age" : 76, "Plan_ID" : 7 }
ObjectId("63954896f323e352a855a8bc"), "Customer_ID" : 11, "Name" : "divya", "age" : 44, "Plan_ID" : 5 }
ObjectId("63954896f323e352a855a8bd"), "Customer_ID" : 12, "Name" : "raju", "age" : 33, "Plan_ID" : 2 }
ObjectId("63954896f323e352a855a8be"), "Customer_ID" : 13, "Name" : "raghu", "age" : 46, "Plan_ID" : 1 }
ObjectId("63954896f323e352a855a8bf"), "Customer_ID" : 14, "Name" : "balu", "age" : 45, "Plan_ID" : 4 }
ObjectId("63954b261e22bf2526524e1b"), "Customer_ID" : 16, "Name" : "zara", "age" : 27, "Plan_ID" : 3 }
ObjectId("63954b261e22bf2526524e1c"), "Customer_ID" : 17, "Name" : "kong", "age" : 74, "Plan_ID" : 3 }
```

ANALYTICAL PURPOSE:

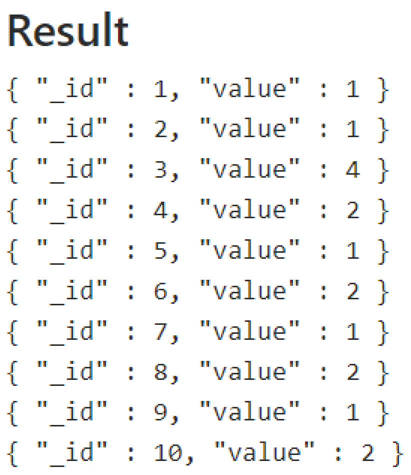
The analytical purpose of this query is to find the customers whose age is greater than 21 and with this we can get to know whether that customer is eligible to buy products like alcohol, tobacco.

QUERY5:

USING MAPREDUCE:

```
var map = function() {emit(this.Plan_ID, 1)};
var reduce = function(key, values) {
 var count = 0;
 values.forEach(function(i) {
count =count+i; });
return count; };
db.customers.mapReduce(map, reduce, {out: "popular_plan"}).find();
```

OUTPUT:



Result

```
{ "_id" : 1, "value" : 1 }
{ "_id" : 2, "value" : 1 }
{ "_id" : 3, "value" : 4 }
{ "_id" : 4, "value" : 2 }
{ "_id" : 5, "value" : 1 }
{ "_id" : 6, "value" : 2 }
{ "_id" : 7, "value" : 1 }
{ "_id" : 8, "value" : 2 }
{ "_id" : 9, "value" : 1 }
{ "_id" : 10, "value" : 2 }
```

ANALYTICAL PURPOSE:

The analytical purpose of this query is to find the popular plan where a greater number of customers are subscribed with. With this we can advice that plan to more customers saying it is popular among and, we can come up with other strategies to improve other plans.

DATABASE ACCESS VIA PYTHON

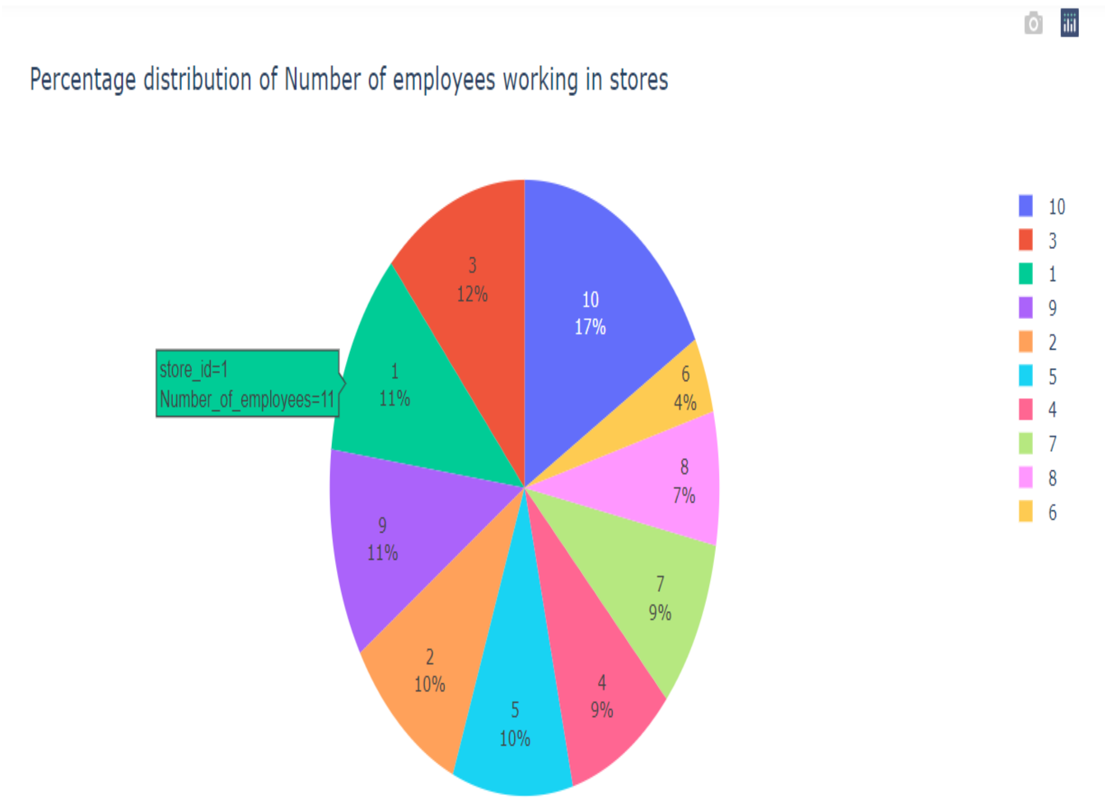
Accessing the MySQL database through Jupyter Notebook:

```
#installing mysql connector
pip install mysql-connector-python

#importing all required libraries
import mysql.connector
from mysql.connector import Error
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt

#connecting mysql workbench
```

```
connection = mysql.connector.connect(host='localhost',
database='store_management_system',
                                     user='root',
                                     password='root')
```

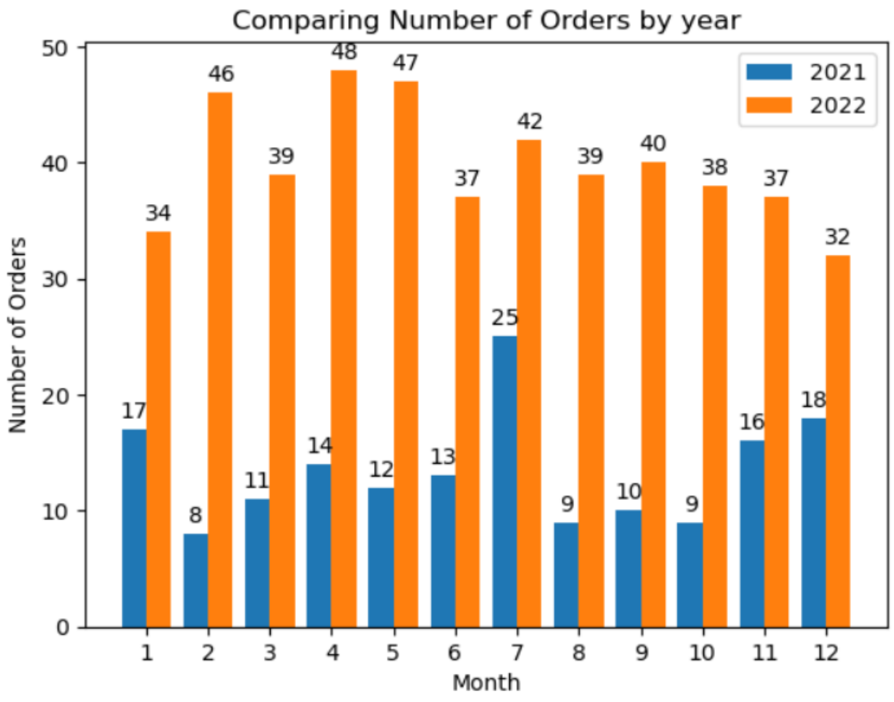
Visualization1:

```
SQL_SELECT_QUERY = 'SELECT  STORE_ID,COUNT(STORE_ID) AS
NUMBER_OF_EMPLOYEES FROM EMPLOYEE GROUP BY STORE_ID ORDER BY
NUMBER_OF_EMPLOYEES DESC' CURSOR = CONNECTION.CURSOR()
CURSOR.EXECUTE(SQL_SELECT_QUERY) RECORDS = CURSOR.FETCHALL()
DF_NO_OF_EMPLOYEES = PD.DATAFRAME(RECORDS, COLUMNS = ['STORE_ID',
'NUMBER_OF_EMPLOYEES'])

FIG = PX.PIE(DF_NO_OF_EMPLOYEES, VALUES = 'NUMBER_OF_EMPLOYEES', NAMES =
'STORE_ID', TITLE = 'PERCENTAGE DISTRIBUTION OF NUMBER OF EMPLOYEES
WORKING IN STORES')
FIG.UPDATE_TRACES(TEXTPOSITION = 'INSIDE', TEXTINFO = 'PERCENT + LABEL' )
FIG.SHOW()
```



Visulaization2:

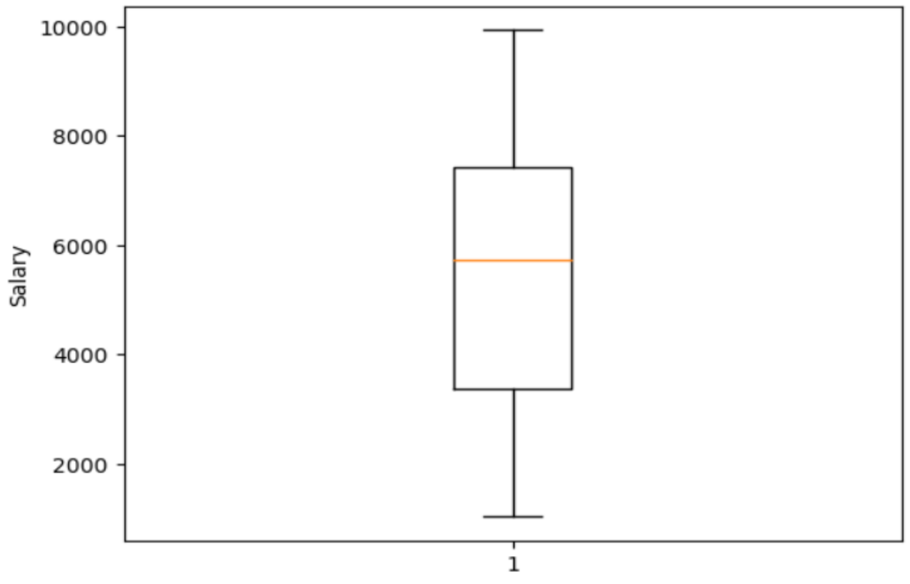
```
SQL_SELECT_QUERY = 'SELECT MONTH(ORDER_DATE) MONTH, COUNT(ORDER_ID) C
FROM ORDERS WHERE YEAR(ORDER_DATE) = 2021 GROUP BY MONTH(ORDER_DATE)
ORDER BY MONTH’
CURSOR = CONNECTION.CURSOR()
CURSOR.EXECUTE(SQL_SELECT_QUERY)
RECORDS = CURSOR.FETCHALL()
DF_ORDERS_2021 = PD.DATAFRAME(RECORDS, COLUMNS = ['MONTH',
'NUMBER_OF_ORDERS’])
SQL_SELECT_QUERY1 = 'SELECT MONTH(ORDER_DATE) MONTH,COUNT(ORDER_ID) C
FROM ORDERS WHERE YEAR(ORDER_DATE) = 2022 GROUP BY MONTH(ORDER_DATE)
ORDER BY MONTH’
CURSOR = CONNECTION.CURSOR()
CURSOR.EXECUTE(SQL_SELECT_QUERY1)
RECORDS = CURSOR.FETCHALL()
DF_ORDERS_2022 = PD.DATAFRAME(RECORDS, COLUMNS = ['MONTH',
'NUMBER_OF_ORDERS'])

X_AXIS = NP.ARANGE(LEN(DF_ORDERS_2022['MONTH']))
PLT1 = PLT.BAR(X_AXIS-0.2, DF_ORDERS_2021['NUMBER_OF_ORDERS’], 0.4,
LABEL='2021')
PLT2=PLT.BAR(X_AXIS+0.2, DF_ORDERS_2022['NUMBER_OF_ORDERS'],0.4,
LABEL='2022')
PLT.XTICKS(X_AXIS,DF_ORDERS_2022['MONTH'])
PLT.XLABEL("MONTH")
PLT.YLABEL("NUMBER OF ORDERS")
PLT.TITLE("COMPARING NUMBER OF ORDERS BY YEAR")
PLT.LEGEND()
PLT.BAR_LABEL(PLT1, PADDING=3)
PLT.BAR_LABEL(PLT2, PADDING=3)
PLT.SHOW()
```



Visualization3:

SQL_SELECT_QUERY = 'SELECT SALARY FROM EMPLOYEE'
CURSOR = CONNECTION.CURSOR()
CURSOR.EXECUTE(SQL_SELECT_QUERY)
RECORDS = CURSOR.FETCHALL()
DF_SALARY = PD.DATAFRAME(RECORDS, COLUMNS = ['SALARY'])

PLT.BOXPLOT(DF_SALARY)
_= PLT.YLABEL('SALARY')
PLT.SHOW()


Salary