

**FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY**

**BACS3183 ADVANCED DATABASE MANAGEMENT**

**Assignment**

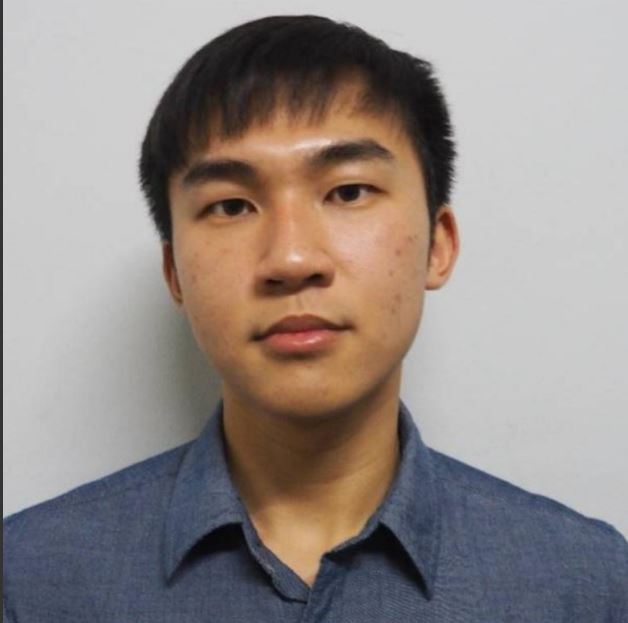
**Semester 202305**

| Programme (Year & Group) | : | RDS2G2 / RDS3G1 |
| --- | --- | --- |
| Tutorial Group | : | 2 |
| Date Submitted | : | 17/09/2023 |

Team members:

| No | Name (Block Letters) | Registration No. | Signature | Marks |
| --- | --- | --- | --- | --- |
| 1 | THONG CHENG HOW | 22WMR03154 | ThongCH |  |
| 2 | RYAN KHO YUEN THIAN | 22WMR04097 | RyanK |  |
| 3 | ONG WENG KAI | 22WMR03309 | WengKai |  |
| 4 | SIM HONG LI | 20WMR10587 | HongLi |  |
| 5 |  |  |  |  |





THONG CHENG HOW RYAN KHO YUEN THIAN ONG WENG KAI SIM HONG LI



**Declaration**

We confirm that we have read and shall comply with all the terms and conditions of TAR University College’s plagiarism policy.

We declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

| Signature | : | ChengHow | RyanK | WengKai | HongLi |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | : | THONG CHENG HOW | RYAN KHO YUEN THIAN | ONG WENG KAI | SIM HONG LI |  |
| Date | : | 17/09/2023 | 17/09/2023 | 17/09/2023 | 17/09/2023 |  |

**Assignment Assessment Form**

**Programme** (Year-Semester-Group):

**Member Name** (Alphabetical order):

|  |  |  | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | |  | | | | | | | | | |
| **Task**  **No.** | | | **Task Descriptions** | | **Weightage** | **Criteria** | **1** | **2** | **3** | **4** | **5** | **Comment** |
| 1  (CLO 1) | | | Entity Relationship Diagram | | 10% | * A complete ER data model in 3rd Normal Form. * All primary keys, foreign keys, relationships and attributes must be clearly shown. |  | | | | |  |
| 2  (CLO 1) | | | Data Definition (DDL) | | 10% | * Relevant integrity constraints to ensure database integrity must be included. * Necessary check constraints and default values to enforce business rules should also be included. |  | | | | |  |
| 3  (CLO 1) | | | Data records | | 5% | * Sufficient quality data records must be created for each table. |  | | | | |  |
| 4  (CLO 3) | | | Queries | | 5% | * Each team member is to design and produce two quality and useful queries for decision making at any two different management levels: strategic, tactical or operational. * Single table queries are not allowed. * Multiple table queries with aggregate functions (where appropriate) must be used. * View(s) is/are ought to be incorporated, where necessary. * These 2 queries cannot be used directly for report body. |  |  |  |  |  |  |
| 5% |  |  |  |  |  |  |
| 5  (CLO 2) | | | Stored Procedures | | 5% | * Each team member is to design and create two stored procedures that cater for the use case scenarios for the system. * Quality, usefulness and importance of the stored procedures must be considered. |  |  |  |  |  |  |
| 5% |  |  |  |  |  |  |
| 6  (CLO 2) | | | Triggers | | 5% | * Each team member is to design and create two triggers that enforces system-wide business rules and policies. * Quality, usefulness and functionality of the triggers must be considered. |  |  |  |  |  |  |
| 5% |  |  |  |  |  |  |
| 7  (CLO 3) | | | Reports | | 10% | * Each team member is to create two procedures to generate two reports (summary, detail and on demand basis reports) for the company. * Parameter value(s) should be passed to the procedure, where necessary. * Cursor must be used in report generation. * Usefulness and presentation of the reports must be considered. |  |  |  |  |  |  |
| 10% |  |  |  |  |  |  |
| 8  (CLO 2) | | | Extra Effort | | 10% | * Indexes, functions, views and/or user-defined exceptions must be incorporated where necessary. * Usefulness and application of each of the above to enhance the efficiency and effectiveness of the information system must be considered. * Linking of all the tasks of every team member in creating a quality information system must be considered. |  |  |  |  |  |  |
| 9  (CLO 3) | | | Presentation & Participation | | 15% | * Run the single script file from Task 2 – 3 to create the new system database on the lab server. * Individual presentation on Task 4 – 8. * Q & A * Actively participate in class discussion. |  |  |  |  |  |  |
| **Assignment Marks / 100** | | | | | |  |  |  |  |  |  |  |

\*CLO 1: Develop the relational database system with the appropriate integrity constraints and security control. (P3, PLO3)

\*CLO 2: Design the solutions to issues pertaining to database efficiency and effectiveness using appropriate techniques. (C4, PLO2)

\*CLO 3: Extract information from the database using efficient SQL query construct. (C4, PLO6)

**Table of Content**

**Chapter 1 Background of the System 9**

**Chapter 2 Entity-Relationship Modeling**

**2.1 Business Rules and Assumptions 10**

**2.2 ERD 12**

**Chapter 3 Data Definition**

**3.1 CREATE TABLE Statements 13**

**3.2 Sample Records (10 sample Records for each table) 18**

**Chapter 4 Queries, Procedures, Triggers and Reports**

**4.1 (Ong Weng Kai) 25**

**4.1.1 Query 1:** This query is to display Average food, environment, service and price rating in each branch.

**4.1.2 Query 2:** The query retrieves the average unit price of a specific ingredient from different suppliers, along with the supplier's name and contact number.

**4.1.3 Procedure 1:** Fetch all available discount at current time

**4.1.4 Procedure 2:** Fetch all customerfeedback remark based on specific rating

**4.1.5 Trigger 1:** Low rating feedback alert system

**4.1.6 Trigger 2:**Check whether new menu item have discount.

**4.1.7 Report 1 :** Fetch customer feedback rating in 4 quarters (1 year) and display the branch, total customers give a rating.

**4.1.8 Report 2:** Employee Report.

**4.2 (Ryan Kho Yuen Thian) 46**

**4.2.1 Query 1: For a given date range, list the Average sales revenues in descending order by Day of the Week (for eg Monday, Tuesday, Wednesday, etc) for each Branch**

**4.2.2 Query 2: List all Ala Carte menu items which currently have discounts grouped by Menu Item Category**

**4.2.3 Trigger 1: Update the Quantity For Ingredients in Branch Inventory when a Sales Order is created at restaurant**

**4.2.4 Trigger 2: Create a PurchaseOrder for ingredient when its QuantityOnHand is less than or equal to its MinQty**

**4.2.5 Procedure 1: Allows a human user to update missing info (price, qty, supplier) in the Purchase Order created by my trigger. After that, the Purchase Order can be dispatched to supplier.**

**4.2.6 Procedure 2: This Stored Procedure processes a delivered purchase order. It increments the Quantity On Hand for the Ingredient that has arrived at the branch. The Status of the Purchase Order will be changed from 'PD' (Pending Delivery) to 'AR' (Arrived)**

**4.2.7 Report 1: On demand & Summary report listing Branches with Below Average Sales Revenues for the specified period of time. The Below Average Revenues are sorted from lowest to highest**

**4.2.8 Report 2: Detailed report showing Suppliers’ Average Turnaround Time (in Days) for Deliveries, sorted from the Shortest Average Turnaround Time to the Longest Average Turnaround Time**

**4.3 (Sim Hong Li) 68**

**4.3.1 Query 1: Ranking of primary delivery companies from each branch**

**4.3.2 Query 2: Ranking least purchase ingredient of fresh food**

**4.3.3 Procedure 1: Add Purchase**

**4.3.4 Procedure 2: Cancel delivery order**

**4.3.5 Trigger 1: Check max of ingredient purchases**

**4.3.6 Trigger 2: Update payment status of cancel delivery order**

**4.3.7 Report 1: Report on monthly total purchases of each branch in a Year**

**4.3.8 Report 2: Report on Number of times delivery company process the order**

**4.4 (Thong Cheng How) 86**

**4.4.1 Query 1: Display all branch transactions and classify it by the highest transaction given the dates by the user input.**

**4.4.2 Query 2: Display all the delivery information given the user input the states the user want to find out**

**4.4.3 Trigger 1: To create a back up table for employee table to store data for the company**

**4.4.4 Trigger 2: To validate the insert statement for the employee back up table if the age is less than 16 years old**

**4.4.5 Procedure 1: To update the sales tax and grand total in the sale order by getting the quantity and the base price from the sales order item table.**

**4.4.6 Procedure 2: To update the order channel status for three different possibilities either DI which is Dine In or DE which is delivery or TA stands for take away. By determining the customer id and the table no.**

**4.4.7 Report 1: Display the longest working employee, just joined employee and employees that worked more than 5 years in the fast food restaurant.**

**4.4.8 Report 2: Displaying the revenue that the branches have generated in the 2 years.**

**Chapter 5 Extra Effort Highlights**

(Note : For each of the sequences, views, indexes, functions and exceptions that you have defined, provide a description and its application in your work. )

**5.1 (Ong Weng Kai) 121**

**5.1.1 Views**

**5.1.2 Sequence**

**5.1.3 User Defined Exceptions**

**5.1.4 Column Formatting**

**5.1.5 Accept and Prompt**

**5.2 (Ryan Kho Yuen Thian) 123**

**5.2.1 Views**

**5.2.2 Indexes**

**5.2.3 Sequences**

**5.2.4 Raise Application Error & Using System Defined Exception**

**5.2.5 Column Formatting**

**5.2.6 Accept and Prompt**

**5.3 (Sim Hong Li) 126**

**5.3.1 Views**

**5.3.2 Indexes**

**5.3.3 User Defined Exceptions**

**5.3.4 Sequence**

**5.3.5 formatting**

**5.4 (Thong Cheng How) 127**

**5.4.1 Views**

**5.4.2 Indexes**

**5.4.3 Defined Exceptions**

**5.4.4 User exception message**

**5.4.5 Sequence**

**5.4.6 accept and prompt**

**5.4.7 formatting**

# 

# 

# 

# 

# 

# Chapter 1 Background of the System

The Fast Food restaurant is a fast food restaurant, which serves western fast-food cuisine. The restaurant has a system that holds a lot of data about all of its branches, employees, suppliers, some of its customers, sales, customer feedback, payments made by customers, delivery companies, menu, ingredients, stock and purchases of ingredients from suppliers. The system aims to keep track of all of its sales orders, purchase orders, stock levels etc in a systematic and concise manner.

The system allows end users to trace, find and manage the restaurant's data for a better management system for the end users. With this system we can use the data to analyze the company's performance on yearly, monthly or weekly basis, depending on the requirements on the system by the management. With this system, managing the stocks in the companies database can be more reliable and secure as sql plus has integrated security features that not all software has integrated with. Security features such as limited access, permission to view has also been a good security feature. Not only that, the admin of the system can give and remove the access of an user.

Another point is that, the management also can handle different branches in the same system, therefore also saving cost, time for the company to earn more revenue. With this system, tracing records, and doing promotions has also been easier and more efficient as with the data stored, the management can use different techniques such as creating queries, trigger, procedure to analyze the data, where the performance of the system can also be measured and state out. Backing up the data for future reference also has been easier because in the modern days, companies will have to create multiple files and store the data, but with this system, users can just use one command to store the data into another table. Therefore the company will not have to worry about missing data or corrupted or usable data.

# Chapter 2 Entity-Relationship Modeling

2.1 Business Rules and Assumptions

**Business Rules:**

1. Each branch can have one or many employees and each employee belongs to one and only one branch.

2. Each employee can handle zero or many sales orders and each sales order can be handled by one and only one employee.

3. Each employee can cook/prepare zero or many ordered food/beverages and each ordered food/beverage to be prepared is assigned to one and only one employee.

4. Each delivery company can provide delivery services for one or many delivery orders and each delivery order can obtain delivery service from one and only one delivery company.

5. Each employee can handle zero or many payments and each payment can be handled by one and only one employee.

6. Each payment type can be used in one or many payments and each payment can be associated with one and only one payment type.

7. Each customer can make one or many sales orders and each sales order can be made by one and only one customer.

8. Each customer can give zero or many customer feedbacks and each customer feedback is given by one and only one customer.

9. Each sales order can have zero or many customer feedbacks and each customer feedback is directed at one and only one sales order.

10. Each sales order can consist of one or many food/beverage items and each food/beverage item can be in one or many sales orders.

11. Each food/beverage item can have zero or many menu discounts and each menu discount can be applied to one and only one food/beverage item.

12. A set meal consists of one or many menu items.

13. Each food/beverage consists of one or many ingredients and each ingredient can be used in one or many food/beverages.

14. Each ingredient can be purchased from one or many suppliers and each supplier can provide one or many ingredients.

15. Each branch has one or many ingredients in its stock and each ingredient can be in the stock of one or many branches.

16. Each sales order may or may not be delivered and each delivery order is associated with one and only one sales order.

17. Each branch can have zero or many customer feedbacks and each customer feedback is directed at one and only one branch.

18. The Operation hours of the restaurant are from 6:30am to 10pm.

19. Food and beverages are regularly checked to ensure that they meet the quality and safety standards by the ministry of health.

20. The management determines which food and beverages are to be sold.

21. Employees must undergo training before serving the actual customers.

22. The restaurant will not share customers’ data with any third parties and will not use customer data illegally.

23. No pets are allowed in the restaurant.

24. When a sales order and at least one corresponding sales order item are created, the

system will find the ingredients that make up the ordered food/beverage(s). If there is sufficient quantity for each ingredient required, the ingredient’s quantity will be decremented in the branch’s inventory.

25. After at least 1 branch inventory record is updated, if the quantity on hand of any ingredient is below or equal to the minimum inventory level, the system will first check whether there is any pending purchase order for the ingredient whose quantity on hand is being updated. If yes, the system will not create a new purchase order. But if there is no pending purchase order for the ingredient, a new purchase order will be created. Since this Purchase Order is auto-created by the system and not by a human, certain field values are defaulted. The SupplierID is defaulted to a ‘dummy’ supplier (SP011) that we have set up. Later on, when the Purchase Order is picked up by a human purchaser, the human is expected to assign a different, real supplier to the Purchase Order. The human user could also set the order quantity and price to more appropriate values.

26. The maximum inventory of each ingredient in each branch is the sum of the ingredient's quantity in the purchase catalog in that branch plus two.

27. The quantity ordered for an ingredient in a new purchase order cannot be over the maximum quantity inventory of the ingredient. If it is over, then the request of purchase will be denied.

28. When the goods for a Purchase Order have arrived, the purchase order Status needs to be updated and the inventory incremented. The user will be prompted for the key fields of the purchase order which are: Branch ID, Supplier ID, Purchase Date and Ingredient ID in order to update the purchase order’s status from ‘Pending Delivery’ to ‘Arrived’.

**Assumptions:**

1. We assume that all branches of the restaurant chain sell the same food and beverage items.

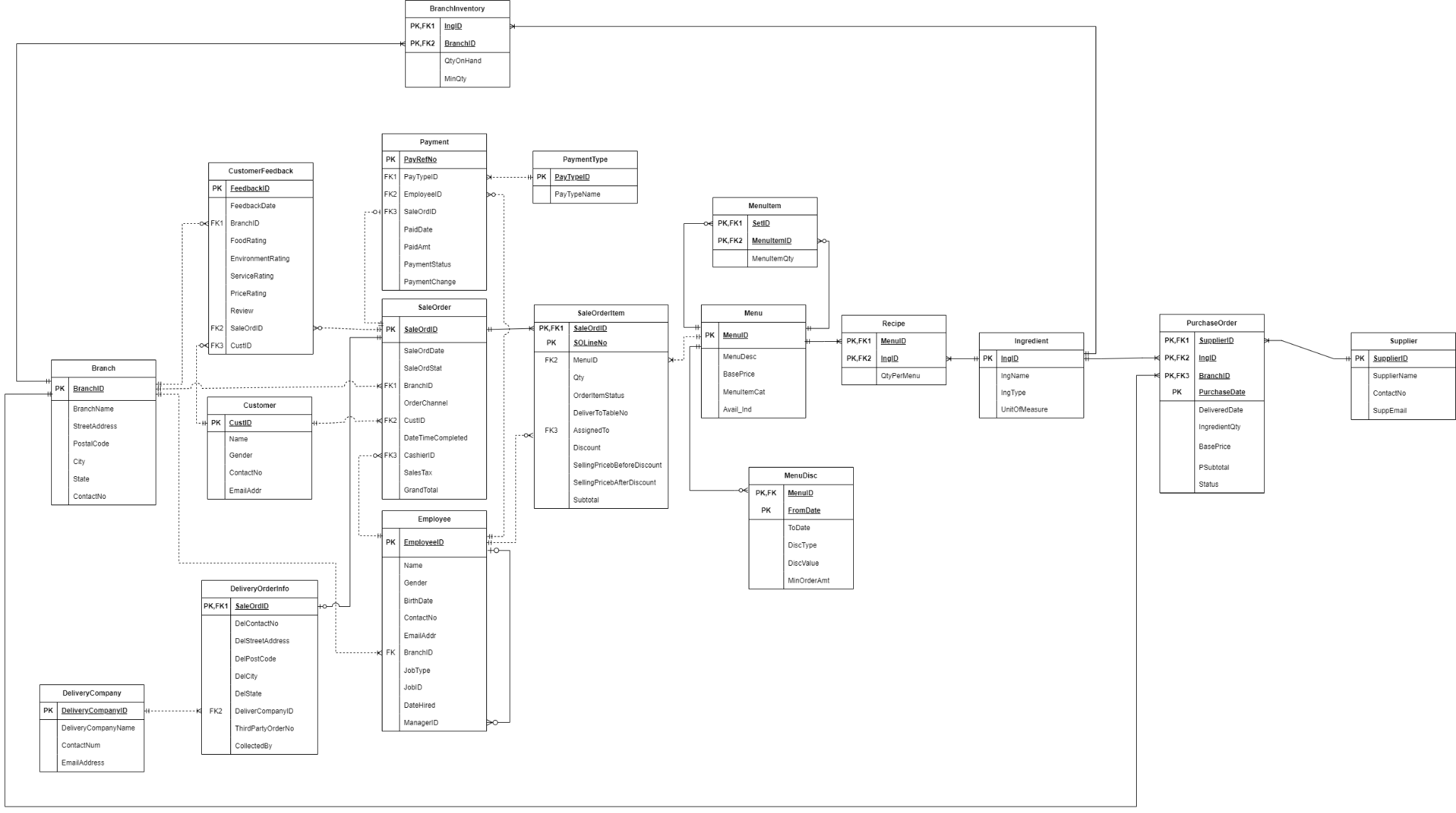
2. We assume that validation is done by the various payment systems such as Visa, Mastercard and financial institutions.

3. We assume that food delivery companies by third parties will arrive on time and deliver the food to the customer on time.

4. We assume that all employees will come to work on time and follow the instructions based on the task given to the employees.

5. We assume that food will not be wasted due to making the wrong order. This will not cause wastage of food to happen.

6. We assume each restaurant has its own inventory limit for each ingredient.

**2.2 ERD** [**Click here for a clearer image**](https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1&page-id=jasQtF-PAra0dogdg8P5&title=ERD%20Diagram%20(official).drawio#R%3Cmxfile%20pages%3D%226%22%3E%3Cdiagram%20name%3D%22oldVersion%22%20id%3D%22ClYMF09uw2yqAIuHjrIf%22%3E7V1dc9u2tv01enSG3x%2BPsRw3qZvEiZ3TnqczrEVbnEqiL0UnVn%2F9JSWClABKAmiQAIE90%2BlYDAXTwAIWuLD23hN7unz9LYue55%2FTWbyYWMbsdWJfTSzLD53i%2F%2BWFze6C6XrVlacsmVXXmgt3yb9xddGorr4ks3h9cGOepos8eT68%2BJCuVvFDfnAtyrL01%2BFtj%2Bni8Lc%2BR08xceHuIVqQV%2F9MZvl8dzVwjeb6xzh5mqPfbBrVvywjdHN1YT2PZumvvUv2h4k9zdI03%2F20fJ3Gi7LzUL%2Fsvnd95F%2FrB8viVU7zBe%2Fux%2F9Nf3v649vjj%2BvQyP%2B5Me4%2BXVSt%2FIwWL9Uf%2FGH5vEg3cVw9dL5BPVE8%2F3P5Yx79XV66XOdRllcDZhvFhWII8ihZxVlxwdx%2BXiyi53WyvX13ZZ4sZn9Em%2FQlRw2hT5ePyWs8%2B74br%2FLeYuj%2BKBorP5aNPxaN31UPU%2F5ztEieVsXPD8VfX%2F7GyyxeF8%2FyR7TOqzvm%2BXJR%2FUh2Ffq74yyPX%2FcuVV33W5wu4zzbFLdU%2F2r71TBWQK4B%2BqtBhRlU1%2BZ7iLDt6mJUIfGpbrsZrOKHarwYxs4ixu70mH0v0Xc5T7Pk33KkFlXP7o%2Fj9vOvZLmIVgWsoxl26TLdzuPteCSLxTRdpOVgr9JVTIx3edMsS5%2Fvo%2BwpzqsLz2myyrf94F4W%2FxU9MzXeuRO3eNZp8dlsPhf%2Flbdn%2BTRdrfOswFXZRlwM76%2B4HOLLPH2uGl3Ej6j9rOr38ue%2F0zxPl0cRcHI6nIdFBQObFgV9gcAmQHB7cxQGxd%2BfJ9Hie7FQRqunxW7Qtutm1Axay8i29nXdv3jH41M1LbrzcbFd%2BubJbBYX0%2Fby1zzJ47vn6KG86VdBHecm7MkZcH649sbHZhyeqrGm05hbixbFErWK8mL%2BvKxma2LM6%2BfsDgPn6Dr%2B6UowHNBKvbv3cl0MerJ6%2BmP3TQ%2FDiysJXl4nR6e3yxU%2FVM0NACAXyISKTIweycQTTSYePQiGWTviWYLa65dJXOqxUp1JfAIDX6Ll8bcBSThEBmRoxxkBcIZwzghFc0aoLWcE1GOlOmcgLWQPBL%2FFq1mc6Usb9ODQjjZMUnQE3hiaN0xLNHGYDPqlNMxBOxz046E8N5AK5WWS5fOr4reOkR74I0A%2FAiDVSiCAwQnAFU4ADJqjNATAZ3dYzwDgB5MUHaflJH%2FIv6Rj5IfBAaIffZASJdDH4PQRCKcPBvlRGvqgHQ6fejyU5wdSYPywjJLF%2B9lslPISfwRoRwBIugACEEgA9Aa43vxvpI54fWMKXhP2zSxGr1vEehYAR6D5v68xZdHqYT4u85M0eNGPUUiREhhlcEZxhDMKgzA5tleKGuJAFxYpPN4Wo1m0df2y%2FXaejNPxxB8K%2BjEBgwcSmKAvJvCFMwGpMV7fiCaDAfeKHvWQKU8WpMz4e%2Fo3vFh0Aot%2BdMJgjwQ66YlObEM0naCGlXyxCKnHQ3WusElFsrQ5fUyyeCY7XwyDAO0IwCalyWL7cJ%2FkC4iwP8QEimRAizZtNFx9kf%2BqDaogHXkzbBbr%2BTCiEHtSFdQoxt4G3bDuClI3HN2rYL8B9gxg0W8nAMoi91dBdjYRHmNvM7gXpXkV5LQ6gKpYd0WrqngVrx9kJxMp4KEfeYCOKAF5CA%2B2Rw4JHckDZMYaBaTMeJuukzxJVxqzB4iQxwFDipDAHoOzh%2FiQe4dB0FSMPhz64VKePkg983OyuosWUQUxLfmDASD68Qdkh5SBP4RH7DtjTBBJO%2F8hBWTdFaRE%2BTl6HS9B8EeAfgxAEWwdz55idMxU9HSSb77Hi6h8K%2F3Q%2FMvuMGpHAaZ1yNTxava%2BLNlQfPzwvUDHffo5Wm0QbzT%2FtIxWs69b8GTlHxwj7ohfk%2Fwv1Fbx83%2FL6%2B%2Fc6tPV695tV4hcigfNNn%2Ftf9j7Vvmx%2Bdr2E%2FreLFrPt7%2F4pPthnb5kD%2FEpFqh25DmirmM3IpNl2cm0i1Ndi%2BLALoEuZtvB%2BRkfPPAJhN6WhLqHfRT%2FWbGTg9swdn979a0GkURDDtaQjTe06xuiIW7QJqXVP9Psn7t58pif3uVMdLPo4EUwfNokEPWrL%2FctCVofYWvKz6Pj0Etpsnh03BZJVB%2BPTj0JYO%2FqklLndiUHl04nuGi30XXB88ld6mDnE%2BEuHXeMkeCcVgcwfNZdQeqe11m6vB9pePjg%2BNCPPcDjKQF7CLfpuPp6PF3weNZdQWqm96nm3AEWz%2BN905boclfD8e5hHs9e1IoYbAMUiaI3S5QebZrC%2FiRK9DIDmwKKTUGHNWV8aqXXplaWfSpX%2FspjOwXmWXx6WjDtFDyuaz9NawMs%2FR4pWqpVvndwxLweju%2F%2BTLf47h5omhsCQqBb0p%2BDDUEywiVMry1s3dqSjOi30QFJppOYqSbJkGKmQidjg8NFP4YBbVOu1xjhMqc3RpmT00LRSeZUk1dImfMqGqVPeHBo6MchFGHs%2BzbiYj2ep0%2FpKloMZSHuaAdG1uML451heJN9%2F7EfID%2FyEQdy8eE2zpKib7cybfvRGK2huOrPs35ihDlp%2FMQukuFRDmgPgyetnxhvyMFxfsRPXICjXLbq26q9w9EHtokHtk8%2BF36%2FV7kLmim2ewKuEw6p4er69k22idp9ZqHhOz%2B1bMmmVj0DkG3b7Tq1bBzCw1r1fVI339X5ObrbGOHB2FGEMhyCoT1INU4XphGQW4y2d4U68z73lwWf1LhPj5qur48M74w%2BPTJkOfrySZ1aI6N%2BPQnAHeOT8vQYa7b169RnwIt2fhkfkhJw91qyE4rwYy5%2FjDkJOK0OkLKg7gpSit6xyZdIZ78lA0L04w%2BG%2BvHAH33xh%2FBDLJ8hs6pq%2FBFQj5bq%2FIEgt2%2BOyLM4zt%2FPZlm8XmtMIfQg0Y5CAlKPBAoZnELE59UMGLRNxTgkoB8u5TmkRdpMi6ldjORM43cQBoToRyAMaQaAQHojEOGJNQMGLVMaAqGd%2Fw71eCjPEKRWOU3yUXrl%2BA%2B%2Bfos%2Fg30WFv%2FeFn%2FaHIb9Lf4MQuToFn%2BfejyUX%2FxJofEuL36jyqs%2F%2Fehrt%2Foj4RtWf5Grv0WbHqK31T9k0BDHtvrXGIfVPyQlwmk5jR%2FyL6nCDMCAAP0YAEL2ZWAARzgDkCLgtWhLLEcOgHSidVeQKt%2FnaBU9xZn8ntdhEKAfB1BoQGIDIqvYrDKu8TA%2By3dQnGOPlVX4BkciO%2FX5aiuyRUdaKNa%2BjuPoGMJlYw31Fh2J0o2gEKQq9ePR58JCy1jvd%2BwBoilDUqm7jTbLeAWFYs4EoIWmTbfLqrdj%2FLdZDEY%2FrbfbDGf7Ib05TJYAtHpF1jMCrZ4FsB03jZY0nNHme%2FwovyQzXAgaA2C0272bBqnqAae8UcJh5xThMWimQUp5cqXyNXpdI5p5AKxiGqScN85kvvJgRkNigeBmCYhFeHCaaZCOQfHpewcLLmimAfCKaZBScfG2cr95HgGtyAERDWkEYpwloBHxAWqmQYqf1ze2Ri8oEOpc9wWyvO0bUaNF%2FDWbyU8kckJGP2IxIfJZBmIRHrhmmgwKqDRvJ7RLgEl%2F8Kk%2Ba7REN0fJ7ErtCAYWCGjIAhC%2BLAMLCI9gM02F45cblAMLmCapR5Ys8H553AGlAglAFPOJzoEwZglIQHwgm2kqHMfcoBxIwDRJMbGywZYBzUoTAQQ0H%2B8cpHIPE8vwb5ylQ1V3IkIS6jYoq9WQ%2BOJdz6m2U8gTsoAOLNlDFDDzuG07hy3xilFAfyni0MOKTGfvvwjsAJtJPQQdmBaptVbLbXn4e3qvNdE%2B8sA0UCHkczujsLedEVoIYYvML%2FTAZMCGNLEHltbVb5p5AJto02opzz4WO89w0QcskNFwxw0uUe7SSwdeER9%2FYOlbBKeZBEAqpnXUI3oVrx9kpxU5QKIhjVAIt%2BKLBHcs%2BNuq3bBUGiYhRqvd1JsXCvWmwqc06o3n4VoHKrTGXDTYxFoKKeUcfvhuyby5MzzG2en9knbyCSpK3mTHohRP%2BsvbYKIlEXa5PNUT%2BrBuadQTu0UI1Ug9qecBbHRNm1RUR%2Blh71k9YYCMftteG%2FJv9qCeMPOKePXEZrCxqqae2JCes%2BkLUk6tSGWsLvfhQaIhjTBor0AjvdGI%2BFwNNoMRVjka8ajHS30aIeXUikbGao8cHiQa0gikJ5WBRiTI1YCUVC15BNKTNjgg5c7LLFo9zOWXuORAiH4k4pCyKJCIABIRn5fB0TwjaT0TgEhMh9Q3t2ff03lUDPFCfjqREzca0gv4TaWgF%2FEJHxwZ85IOuUyA6bTpC1L3nEbreTKGsmZyQkZDZoFEpTIwiwRZJBzNM5U6kKm07gu0Ou1hoTyJv0%2BW8TRdPi%2FiPJ4Bw3SBjn4M41JkLB1ndoouQQ0HUREkvrhnp3Bki29w0SkLcjd7GCZpwxs8rCELb4hTsgoPC6M4l6wCv9%2BphPN%2Bc1W4pAB9FS%2BKYco2W8np0%2BrxeG01TUMuMAAVnRgSq7LVNhcIzPLbhbngjaXbj7Pwcz053hJz0Y6E%2FoDQkqbgpuhJuQ4Tej6VdjuZZPlupKTZRx01yY5L7ek58oIBMq%2BHA6zDXhwss%2FzVHnZ2aYm8GJpdSOlY%2FDHCcG4nt5NrVlFeIRXg6cs6l59U5MCHhiQChlkZSKQl7mJgEkE0pqNf1qU%2F4lGeQTxS4b2Ki4Es5vpD%2FuW47KQBjzB7ZjXiEY%2BULIFHhueRtsCLoYmEQfRUjUg8%2BgFTn0hIzbMgkrs8i%2BP8%2FWyWxeu1xmTCgBQNyQQcslKQSQvkhiaTMaZkpV4DOtlfFWULUsMs2OI2XRezZDbKrCE9oEBDJgBHqxRM0BIrMTQTMAiV42OCTnZVNZnAb7Grxotpkm%2BUZgFm16lGLOBTuE6BBXpngba4hoFZwGeQGUfHAj6961F9FiBVxK16NNIUgj1AQEMaYMgwCjTQHw044mmAFAjFh7fxZAL6UunqMwEpAVbBEGX0WrTajNPc1AMUNGQEhmShwAj9MYIvnhEYhMLx0QF9wXT16YDUAe%2FnSTa7LQZ1Fx03TpNSD1jQjw%2BQTA18IJQP6hLv4vggICXD6xtHHUqokQ6UYAakKFgM06J46Hh2qfSpAQMKNCQDiL6WggzEh18HpHB4feMqRAYQWd30BSkNfozK%2FDBfi95RnA0gWvpE51A4CfmWZxaRvch4F4bBZD%2BDUek0PpXDqPxwG2dJ0cHbzDBbWho4rxECrjx5jRBCcYGLOa8R1pCJN9R31eag1Ve5TSVUyeen90TY%2Fkf9REKhdThiFy3aZmsyx3ql62H%2FAi5Iup0sSwRGQC%2BkSVO7OSDFT51qNweQ1bPuCxQxfnxhl%2F9cdMBMQgzQ0S%2BrZwj%2BSv5KCTu%2FiK%2FhHJJKquA1ZLjY3ZB%2Bq6g%2BubS6L%2FfJ5Uu0HKURc3iwaEgn4NOUgU7E13IOGeK4laOTThZORemE1F5RQqGXpc4swuzt1IlFwNspA4tIUMo5HKO3k9cS0cn6qSiNkOrnh2WULCCbEANKtCOS2msy3MlvAR%2Fmg190gDuhLz9THxabE6bDYk7Hu7Wmev6AFwWxSHPAG2DlQvCzO9rz3RAvO4I31PP5rmWQ0m2Z8LloODu9W5rodrDroWICaKgcg6wQ036yG%2FS1tbEMBsFV600uAxs2c2I8J7uWQSquGp3sNvMAtrmWQcqn48jhP9xxLgteNNzwQvpM7spJB1IRfpxrGWPMnslrhYDa8k1fkFKq3ue3LOjQkD%2FAbioDfwg%2Fv7UMUnDVhz%2FAa1r3Bcr7t4eD3%2BLV7ITEpAGDgKH0BGDAUCoDg4g%2Fuy0eQV8KMekHTH0KIfVNqCnGBBENWQR8pFKwSFuU%2BMAsMkYjKfUaAE7Rpi9IxbK2%2BIyRJnrAgIY8QOEEldWaU3%2BgzOOw50MqqGGePqWraLHvQeKVmwGdoJ0376D62dKYdzykjCEnBxF6T2vf8fGWCCPQEf9OgZpos3dbRfvHHxnLA3FhV%2BURaRNHXAQH9xc%2F7B6B80xrLx9fuomu43j2d%2FTwz%2Bkd2ER7V5FPf%2BzX33aJQbfVeuPM9KpGL%2FhJ4ypC%2Bo%2BmrqJ6HsDOulna9nLgVUs6OIu6YUa%2FnTjiESAWjooMO7GIdxZZ%2BhZ5byYBsIpltWRWrVjlaqSluoZHiYY8Ag5VGXhEvMPIIvXe6xvzKBaUoxIwqTZ9Qaq84wh7kAMfGpII2FRlIBEJTEaWxj5VC3yqdV%2Bg37m%2Fm0jT2fcoL1ZsnWkEvKonQANeVSloRLzLyGZQN6WhEdoRsemPNtXnCVK9%2FLD6mWTpaln04njpogcsaMgH4DqVgg8C8XygsuvUBtdp0xekCnkXZz%2BTh1gDLgDn6YnOgRykMnBBHYAqkAvGmIOUeg2AJKNNX7SUWMr0YAJIInq8c1CeR2ACsUzgCGcCh0EvHB0T1DAHJrAcUg6cpstSIVKZBRgQoCELQEl5KViAtiZrjyzQVlJeJSLoVFJeUSIgdcDLLFo9zMfpTeoBAhoywTjKyVehy%2BZkP3TZkrqsQJ2h4mxkcm2PkyYy2UfwrHPMhxg2aSOTA7wlF2%2Bp78oCDoX2KSvC%2BwnO5wRw36LFN1qBpcG362GoNL2O%2BPbwliy8pd7xTaHonk%2FHwIbtHYJpod3rOiwNpgJ0zFpHTXeElIuUmToU3w%2BoIMWazME0jzzysScjvlA%2FWr%2FpHBxSy%2F6azeJsOo%2BKreiCgLveqRxC63BJstHR6Lm3TJyZ%2Bb1lok0A6A0cEznUs2JEiRxcUnTWKZFDPQ9AjbBcUpbeX9Tl1yQGTObAgBv9JAwXxGz%2BYjY7uYhP5uAyOF6lEbJ5rRCgczd9Qerc%2B8yieckYBqRoyCWDy%2BF61NitdzDn03Qiu4I8wg5qpyI6p3ORXawhe%2Bgiuy4phd9Fi3i7OH7K4%2BXpbdO4hJS21fIoflnyZB4qYC6tkas2pPSw74EEAvQb4C6MSW8olUdoabH%2F3hSdKj4rDY3cwjyPz0wMpk2xx3VrQ9PaEDsb9Bzk2q%2BG0jI8ZF4PB3h%2Frlt8d8c0zQ2CIUgxQC%2FjD0I04kUXj5RxR6Hoc1ovPPodpPoUQwqxd1%2FLDb38pbEGZBgGxGjIMJC0QLJXGfG5NT1Szh2FRZ3XetEpwYGiDEPKs5%2Fj1UupXcn%2FFiMHRjTkFEh%2BIBmnSJBq0xtjHgReq0WnNAmKMgqpk37LNzpTCXP2BI2oBMVrApVwtBp14A%2FxOTZ9lbMn1DAHgrBQlBNuJirfOe7yKH9Zj5EsekCChmwAxlMp2EB8hk1%2FjM5T6jWgk7VUUTYgtcireJEUXXOfyn%2FKMRAKNGQCCtsoMEHvTCBBfk2fFBuvb2yFyMCjHhT1yYDUEN%2Bv18UaHM%2Fu1eYCehBoyAUURZxkDQ6oP9AFB5xPK0EiijZiwERjdz69CPLsSRMx4GNGf8%2FEcEidPQdryMUb4pQJwrMPf48VhCefC7%2FfDVxsLvWRBgIxueSZfLpOJjRx34WhfzB5A8s8PX2LD7dxlhQdvA3E2G73OMXt0M5BRAnSzEHXxyDdNYMVnuHHGTrBT0BKzcj0cHSTMcKAnaMg7Z75hDo4p77I%2F4UgIPXj06Om66shi%2BUgoIeGNBE5AakWj8IozSnsuJ4HEKBuBaRcPB4b24BpTxgwo1%2BoesBQ1UtrYmHRHNmJRXwETsAgPkujN%2FJaIegLy6vPKqT0jFjlKl4%2FyM4rcqBEQx6B7AEy8Ij4OJuAQrhWlkc6ZQpQk0cQEvfLRETreFs9TmcSYc4MoBGJhJAZQAYSkSCwJiT1zmvRMhdH%2F0NIPybqEwWpaKIXjmmUy69kDYQDDdkAovilYAPxYTIhg0g5Pi7oFJmvKBeQIuT7n1Gy%2BN%2Bn1UxpHmCOvNeJBwYvvfUGw9yF8c4wkE%2BuKjAXTnrPqdtm3iHBR%2B3XQUdm5x07KGeENI4dHwkPyL3hY5Clds1hDTl4Q307dkJSUt3bGJ%2FeCk10N%2B0EKKOcuHJFIYMSqvX%2BlUU%2FC5nLIwv37NQl3DT17NTzADa4tkEqnKMSOwa07TDARrs9sW2Q%2Bihwy1u1EXZuEW7bsQ2G7AHSSCN8VohmEgCx2MZxM2hBLHo7d1iAoiGVUIir%2FCo%2Fd5FW3hoKdaDGnI5h5BXwhDYv5wUUdAQkjYASOtibdNcK1KaBteTjLXGKO8QrIoVGcPLB8Ps9f4C4Q9s4kXV29ZQVtBmvlFJ1eqqdZJpYKB19fBZeKYvjNgyyw9Lvx7swOH2uAHlkH1K%2F1a54UjMxINOHbZCCsFoBW8NjRrvEIDbK3QBM00v1pA5MI14EMklVuWIa0WLQgExTTwxgGtsk5eFPq59ANN0hoyHRQF5ayV5phAd32eYYM9TyWiwggW3TF6Rm%2FC3f3J6p%2Fy0JwciBEQ0JBdLbSkYo4gO9bJNBOFWOUSALbtMXpFj6Y5XkXx8%2FF1PoJRtlyPDwMNGQVEhNtXjTLXo0zTaQ25A8PHPwXLAo%2FbS45IY2whJsDPgZpZuJMaITM6tdx5ReweRkU6rnAfjZbLQEqKpgDg8Z%2FZxtFiiY9C%2Bc%2FfGK%2BPMxa4yyJa8VopNsqSipkLJlRSpfouUoXzGHB4mGNAK6pQw0Iv70y2qryqUPkXRSKxUlElKtrIjkfvMcy%2F%2BGIgdMNKQSSAkgA5VIcO6FWtbylQQSAjQ4ICXPabouc%2FmUx1868wiE%2F59ADamOAo8I4BHhqRFte4zx%2F7QjUsMciMK2SQ3zW775uvoYKZ0akQUDGvIAY%2By%2B0NSIKE9hlRXRsCZ01YTfGYYzOYj8t0x%2Fcib2v%2FyEJ0QUlSSxDmiTJsbfNLAYah9HL3WQPx6N7eEt9Zwm0bYp5Flp5kAJZmwemA7dPGBMD8oX7PVB83mwo6AaecBuubzAbruiwU5K0Ad2t1J%2BPLob0NTy5mFpSOqMsMJSg9o2VMvi73iz6cPGpXG82aQgrJPjzYaCWXVfoFVqrGdKA7reGGCj3zuhA0W0%2BGuD7Nwi3vXmMIjE0kiDnFYIh36bqD6xkBLxHrFo7nxjAIqGVELhm5VHWjmQVU6n%2BeRSc4WEGLWagjYw59UUZKeRR03xsAiywMSQSa2m%2BFhLPt5S32qKQ8rnt8Wzz6N1%2FDWbxdnpfZN%2BSgoePGgaJuUup8foQQfsufy1FIe%2B1r00WopDaqM6aSkO%2BHObvmhJvVou7MWqDkJKN8xouPsFsy5%2FIYWdWMQLKQiQWgop4NVtcNCSnqBilaviN8vOK3KgRD8ecRl0WOCR3nhEfPygO0avLqcVop4EwCO2S8qqiEfu8hOVfdXnEQaUaMgjFGZf4JHeeUSC4EGXlDzFF5UZkEvox0x9LiFFz8ssWj3M5de55ECIhjwC5lMpeER88KBLKpziS8ZwjB90wV5a9wWSUffG%2Bu7l%2BXmRxNk4qaIHEOjHBR4paSJUnOaECZgfLNo4Eru%2FFdwDbZG%2F%2BcGjB4c05geP1BZ1Mj%2FU8wCY3vZIeXE8TD%2Bg%2BYEBMxpuDEBs5P%2BSyE4s4s0PHoO9UpoXRF4rBAiNTV%2BQQiNiFc1DSBhQoiGPgNgoA4%2BINz94DG5K5XgEdMi6L5BosoeDq3j9kCXPZcyYzjQCOuUJ1EBQuww0IoH3wWdQPFXjEZ9%2BwNTnEVLwnJYz%2FSH%2FkmrMIgwQ0ZBFGOpAAYv0xyLinQ8%2Bg7wpDYtQrwEO9YioTxOkfPlhGSWL97NZFq%2FXY2SKHmCgIRUw1J0HKuiPCgLxVMCgUI6PCujzlqlPBaQCeZdncZzrwAX0ONCPC9B6A1wglAss2mRC%2FXFBwCAzjo4LapgDF9iodsO%2BXS4t5m8xWLNRnmX3AAINiQAKxktBBI54IhhjxXjqNQBKwjd9QQqB0yTfKE0BUOz9ROdANkkpKIA6tqY%2FCmBQCMdHAZAssukLUgAs07Co%2FRoAiR9PdA4kfpSBA2xDOAcg26yaHACpHZuBbivD%2FrJCIFKVBSBt4wlItEiE%2B8UBruLjmdpGGCLf5j8jcfT2sPmA1kJae017WNZB56Mn%2BA4mxHrqjCiAPmxJynhTdKr4xGo0YfTME%2FnMxGDaEnhcCYCmtUHW%2FyPFYZSpITA8Zl4PR3h%2Fslt8NxE0zQ0CIpAT6ZjmbMIeXkwjPqI%2BbK9TMxWfem1IpukkQCrKNKQAWRVoBKLpBhkNiQY0S8leaYSH3Ndv3qNSL3ktFp3ETSX5xTFIcfPbOC0Ow4NDOyZxDFL3JKCyX%2FS3WJXn6VO6ihZjKPhbfxi24C%2FKNXe%2B4C9KVytPwV8Hgb9OAhBisKSu%2BOviTZl4U0dK%2FhaoiDZ7t1Vbh%2BMPXecZQb%2FJNt3Tz0Z8w%2FQPvlH8sHsKztONQouWpsa2eTDlrH6KbL9hmqEs7OenGVKt5Jlmbojhz%2BhcWNvDmgoHLqztGIyF48VxyDtj23QDat8MT6O6%2BHAbZ0nRR9tTvDciFrE%2BBWKlqwRv40d7flfA4oeEPt5SX7TgVTssalYIwkFIgTx7%2BByvXkpJ6CpZPxBzCc6diQ07KraHxo02psDC10qO7%2BNwGNCnRtPMmvEcOztG22GAuT0NOIoN1Q4DmokBYo1jkIcBaOmH04BumNFQw4HTgD6PnTswjfBjZwftaw6YRvSBwJAkAycCDRbIE4HrLF2OoYr9kBQDxwQnIERxTAAUMxnwZUb8gbM5xgL3nBaLej4AvTgmKf%2Fep2MgFznwoSGZQLUpychEfKZ3xxxj5Sleq0WnylOKsgkpkZaHIPebZ635hLnqlE58AlWn6PmkRxIRnujdMcdYdop6DehUV0pNlkBLD8YS%2F9l%2BHCFN9IAB%2FXjAInVO4AEBPCA8y7uDBl5JHmCwx6jPA6QG%2BTlZbTN2vF8ez9ihABN0MElpxASMjtS326z%2FjbO0qy31ovSlTg5sqVYwGc6WaqH82GdtqbUPQxpbaoAOupHzz8XgRutKDbGGfLyhvl3UFqmKXhZoWcRfHx%2BLQT65kZmACbQM4fDwIQzoNiJ4WArHjQiYQHuVuq0RmkCtoxkh9Mk91EwM2MQ6Fqlf7lZ%2BsIB2Q4yGG16wgPZqAWXnGfEWUNSyzpmHmokBPOPYpD4KwQZvw4x%2BTGNTHLbyC%2FZ9i6xyKKnIFr3OILpYsokuZo08PM0vcyywjbXk4y3xigU2ncNf5AZnMkTgX%2FCrz8f%2FFPeNXwiq7BD9RhvbFFtFebRR851R9MqBNurap6dyizbaNSHMMNM7lG1640Ja9%2Blt4ZIc5fTugPbb%2BMen59idvv6Vf%2Fn95n9Pnz4u%2F3chXYVPTmdC7OKohS1%2FtktuWUaf23%2F1l%2FXPTb6x%2FvttGnxIN4vwx6ePNBhgW8l26xXtQtZx6dk%2FHLIOtzKG77GvgGe3ZJySX9Eue%2FKl5PHCgxlidV30LKwhorQBpy2NjW1QLMc6%2BVz4%2Fabjc91utE4%2Bi22zweNtgW167u0zDmdZmUKs8z5jsNcIEy2oZyec6co240Js%2F0skY6HeZhhYSyZlCizWKRdg2xmrqhx77MHw%2By9M8%2BALb55z%2F%2Fn9%2B4%2F3v9%2B8%2FyeI%2F%2Frz582r9%2FVTPOyc65KOiwMlYi8FxWLGPFk5EZ5Py3i260o2AQM8CV298Wd2TxDp7PCWOM1ALEEX6v1jz0VkeXTelNBrUgbOp%2Fn%2B7Vn0PP%2BczuLyjv8H%3C%2Fdiagram%3E%3Cdiagram%20name%3D%22suggestedChanges%22%20id%3D%22dpHIU2U41dhEEFn-8v9d%22%3E7V1hc5vIsv01%2BuiUACGhj7Ed72a9SXzjvPfu%2B%2FSKNWOLehK4EE6s%2FPo7CAZLDJJBDJruoav2VkVcWSPB6Z4zc7rPjJyr1esfif%2B8%2BBIHbDmyx09JGIyc65FtW%2Fx%2F%2FMKz%2F8T2LmTvuA9%2Fi4vj4upLGLD13hvTOF6m4fP%2BxYc4ithDunfNT5L41%2F7bHuOl%2FDXuH%2Fwlk67%2BTxiki%2Fyq547frv%2FJwqeFGMgaF%2F%2FPyhdvLi6sF34Q%2F9q55Hwa2dOR7bz6I%2BdylF3b%2F8%2B5SuI4ffdt4s2r1yu2zO6suK%2B%2F1xfp81frJnlc3aZ%2FWS%2FR6%2FSvi%2FzL3Cj%2B1PIuJSxKzz66nY%2F%2B01%2B%2BFE%2Ft0%2Bp5GW8YK%2B58uhGPkz%2BE5%2Byfqf9PdulynfpJWoDM4SNectikfhixhF%2Bwtq%2BXS%2F95HW7fnl9ZhMvgb38Tv6Tig8Sry8fwlQXfc4xl7%2BVw%2B5t%2FWPYy%2B%2FBH%2FuH3xZfJ%2Fm9%2FGT5F%2FN8P%2FK5lI14mbM2%2Fy9%2F%2BOi3esUhXy%2BKfxY9kScpeK9h8565bTdBWvfF%2FsHjF0mTDXxfDObMCyZv8dRmSv94Cw%2FKKa4udoHCc4qJfxN5T%2Bdmt0MDfWwBCLX4cCT%2FHcfM9C%2BPLRZyEvzO0LIunu4ul7etf4WrpRzw%2F%2BEHl0mUcbAQmwuXyKl7GGeCiOGIS5rI3BUn8%2FMNPnlhaXHiOwyjd3k33kv%2FHf%2B7V%2BIM7cvl3veKvrbfX%2FL%2Fs7Ul6FUfrNOHYzj6DcYj9YhnMLtP4ufjQJXsUn58UTy%2F79z9xmsarzii0O6KwQJ3TFHSwMTeRMHd3exB1%2FN6mob%2F8zuczP3pa5hjZTm%2F%2BG0ZqgFT7aMvHWX3O1ewU86f8uNxOWYswCBjPVJe%2FFmHK7p%2F9h%2BxNv%2Fh8ripHOR3RsQMHpyUa2oz6Nvbbw2g9uL%2Fk2T7yU54GXqJg3QqRe7gsf79uNLsHZ%2BDP15pRLebY%2FL2Xa47dMHr6O%2F%2FLaQX2LjbYv44OJkVBUbWEQaPRzYuDKTGJRkxiDIFJTM1gErPmmDtPxmVBKD7v7DRiSjQCcfr0JCh%2F9VeHF%2FFACAQqgBNhAIX4OREGPIRhbgZhENt2xBjKCCTGgDR%2FWpaE5T9YFLCESINCjBNpgAV6WfAi1gCWNVi2IbShhUwGhjac%2FvStro%2BfiIFWtMoC22WYpItr%2FiMxcgONQKbZHxayZbGNZn%2B4s78AKfbZv4W0BWb2721BlccgkQOsKVTWzK6y1PiQfo0xkgO4OCfuAAv4ssJG3AEudxA8ATt3aKFygeEOHZKmR%2BQAcY60xxJaP638cPkxCFCqChqBTLM%2FLGTLYhnN%2FmBn%2F%2BbtNbBnfwGyHdTdQGp1GJ97UWV37ccifqAVz7IOdpn40cMCV6sDPtgTm4AVB7LCRmwCLpuYGMImWqhfBuwl5EFGXAFrjpRVszsOSj7czcv2r9MQZ3%2BDRkQTDYAF8Rb9Z0QDtNMA4eGBnQa0ELKGsKUwI5qAOYfKAtlf8T%2B0n9Av5olIgAoC8QuJSGAgEo7wuUNOJJwWmpgB%2Bwn50yCigDVHylpa1tDwZ5iwADpZAAZkmv1BIXsmIztX1Y5zgMp8Txae76B9WmxIFrP4hTX2mk3j5cYB0Hl8RiaezdhjNxzOlNFHI9jjjGw868KQ%2BCXWWVhWVzHWtpzdxrMz7ImNwoqDFt2Og2YT3faiVLEJU5w8ycpTikGiElhTqKzQ5lSCLD1VA53IAyzkk6knJvJgiKun%2BMpEHt5ikMgD0hTqyarsfZowlmZNuGy9Jv6gEuvEH2CBX5bCiD%2FA5Q%2Bm%2BHt6GP09e0uqXld0EIHQCuYaYS7mCZEDMqDdB6VAJ%2FYAC%2FlkD4qKPRjiD%2Bq1UMzAsIcOWZMMQFEnyRoD0DDdYCQGGjFMMz8sUJO5J6qZ3xB3T29Y7p4euXtiTpJC7dtTFgZ3JkhnENPUDwvV5OyJaeo3xdpz3kKpMmDqn5NxJ%2BokKQtcqE%2F90Ahkmv5hIZusOFFN%2F4Z4cc5lqUm7s3e%2FBIDcOFGnSVmj%2BuJH%2FhNL4Lc3AgMyEQBYyG7QOcaCJyYaZPlkuoif4shffnq7mrfQ5rO%2FZY%2F26ktYFHxMki38Pn3nAP8R88jZCMrw9n%2Bt%2FCj4tsV%2Fkt1bJmgDew3Tf2f%2F%2FjAeW8Xr%2F92%2Bnk2mxevr1503Xwt2wYGcbP69%2ByL%2FO1e8fPuz7Svxd4G%2FXmyHF8PdsSTkT2trC%2FLGA7K70i52ivet45fkgR1jC8VyIxWs59AbVZ4HYQlbrl38lxcTtvTT8Of%2BD1YXk3cZpXtLDva0QGXVMkyMmN%2FC4q9ax%2BDBcZ3KuJNq%2FOdPpOu4HPT%2BZudTC0J70t078lvcvd9y4U5P%2FsqHBxE36JyDTBy3w83nf5fff9RZWxZs7%2FzNihWwJNMjRSSlano0t5xm669yoQZ1AUadhs3W%2Fd1wOO%2BqlpllelTSCXI92otD2hdAOg9bY1k%2B5RPxd%2FYIXxjQanvUGfe0jQAsEFoIuoMmFN2EBFWEwhDfI2ssK7M3t9ZB5A3gQI8iEolSoM2ksiT7afW8jDeMwRcbkEOfWAWwWKCWREyswhBDJGssy703t4eRB6Y%2BocfMSl2LuBOpLPPe%2BZsfm2cEnAIZ0olDAIM%2BNTdi4hCmmCJZY1liu7l1hr01QR2QqFOpaD7abYH0l%2BxbEsBnEciRT6wCWChQ3yQqVmGIWZJltdDZwOxLdEicFnVO4s6TskB354fB9eCME7ojmSgAMGhT7yQqCmCIa5JltRDFjKAA1DuJO0%2FKalpGAT6uDjcrGMoAqHfSMGS3OHaPGIB2BmCKeZJltZC0jGAAM2IAqPOkLIUV7YqZg%2BLQWEBXMBMLgIVue%2Fx%2BNlZnofCbJXFrDwVVTgjlZ5QmDO7omAVDfh%2F6c0poapSgtCgelFOCKOzq3Rmh0gntOJP9gTFZI4hnJXihe7qhQONBLjzH63BjjLAtsGxZNS2IQFaKeHztNCLvgsN3v4F3gTWeTpotf0QZdz%2BMod33s2u%2B3xA4RQuZd9A7Cx1jrbODkmEGDXaNXDtog4YiEmnvAW0mrTnPEUvng1aLhu7Ip40KYKFA3ZTnkCuUkQpTTBpsWQDWnHm19lLa1EuJO40e7KW8ZusH6JwCGdaJQwADfwPpeVfs4PgI08337aZ4HJ3LM1roHdaou96xbzqtV%2B8o94UaKB4KZxlIisdUsKJyX3023R%2ByJwnEtSoDz9VKIPBCXVbti24%2FlhxfN4xoq75RiAlIu%2FtW3fak4UY9dJdhS0yItOLseRu7q52mYdvYTo3OOOxt7PyJ0aIT62TsyIIgytb7829jd0Y%2BLUGBhYKsURKp6GMbWxGpMGUb22nR7zmAbew8DIlRoE2jshpYMAqszfmAsU4cAhj4W0iSxCH0cwhTnIWdFh2jQ%2BAQU%2BIQqNOorAYWHAJrYx9grBOHAAZ%2BOkMRFYcwxllYqH5EInbikEgE1jw6kcW6y8SPHhbwlQ1kQCcGAQz5sqhHDAIygzDFRXjSQkIbwsEEeSAShUCbSGVpblumeLXwOVKX0GkEcvATrQAWDdTnh4tWmOJMPJFVtaGfxjyhZj%2FcuVQW7K789SJkCfzdCeTIJ1oBLBToIEVUtMIYu%2BNJ3UmKA9%2BwoJMUUedSMYXsQDqruvwRrthVvHpespQFRC96jQCiF8BCosGJijhtlE9xEtizIsjvAwAbZZWrWUimAq4oqRB9sKJnpWdPgWllXLs6LiJX5WnFH6EXV%2BXqIJOiCGvIpsqurCJfsyWPnWSzFUE%2BR4%2Fx8YXaiPwaDj%2BDGr%2BGStTyRzCXSIRdl96kvAJtqeVSb2WzNX5HILpdNbfDhg31wIOOuxqX1lv%2B4GDJB%2BevbXQVNllqXdUMdVFzsMkSl3xwftuGzsh%2F3Qcere91hwK1XJ5FPlBFLWpsG3BSC1nB1a8eaG2YcBV2XRKp0IBoWYi9elmn8BkFMpgTgwCGe2q4RMUgakwbUDIIwYTgdEvopQ8K%2By2JPmiAsyy0XjOOR54hH9Kvh6WCYZIIRT2XRCKgoF9Wy4hEACYRda4NOFkEuJ5LrSxi2hUfxCL0wllW7DiLuE8TxtKPQZCw9ZqYhFLAE5MAFgHUZomLSdREDk4m0UJJA8MkumROhVWnRBU04FVW4DhVuIvXPLcEDCNL0AlmogHA0E1tkbhoQI3bAk4a0EIOM4IGKOx5JBpwfrzOxnU04CpMN0OjAIpaF4kCQEF2g9ZFogBwKECdMwJKCjBrIWaZQAFmXTu4iALoxausgW1FA6QHTulEMnEAYNBucTogcQAAHEA08aLnALIMdXPrmE0DJkQDUOdKWcAqHAsy8yM%2F2uBsZdCJaKIDwCDe4qA%2FogMA6IDYnURPB1rIUUZwgRlxAdSJUlaxfizCJLjj2Mz9i3C2JOiENJEBWBgXijORARxkwBETP3Yy4MnC1M3txGg%2BkMca8QG0uVKWtDjalvweseByaJUCncFMTAAYuslfERcTMMVg0ZPlqZtb12wmQN6JuHOlLGz96Wd27N84HIZHBcgP0TB4N%2Bji2j3vgOMjTDfft774cYTlzIPxh%2FncG%2B2ee5A1wh47%2BSB7cceSkD%2Bmrfn4lpOAPA1B5QQD6jQEEX1VbaTv0xAq41rVcbudhgAvA9T2xW2N%2Bwsd%2FPjaoLIOINv%2B4zPf3N6H10WN5Fd7Plo5Q4El99SC1mxV2RGHXldh5rBrP8qD%2BTxZu8tc%2B7Xycr1%2B%2FR6dy4d6ShaWlYenZPilaXp9%2BztHAJ3LBywkqLntLFvWqshFjW8%2FSnIxl4VAzZlXq1%2FeXGHrGzELDXCubX3bZRZf%2FRXKLjjAmCcuASwIqEkOFZeocfDHySVaWDUOgUso7J8jLqEBzrJ0KOz7X1ZEIZRCnSgEMOxTYx0qClHn34%2BTQ2BsrOsxsSrsuyMOoQHOsnb3aeWHS%2FLu7wPsxCJAob8s4T5f8R2Pgta1d6KGbrRbP%2BeOjlbPiXo9a9SqXu%2BsFXZlwcr7NXYqrV4g1dh5wn6ooEnVEqOeSuzmlWEvquOaVWJnj2XxMjsql3%2B95PiqYUS1dY3CS5Ruus4%2BribjeTPaX14EyvvtcQsdctALzm5ALEKViuvK%2ByELhsMurisikZacaCdjWf3DcW691oq67rCnxSewOKCD486xha2MURhSUWePG3QcDmcHuwhDohNo06isBFIJnXKQE3kAhnrq9UNFHgwpobPHsl44aPJAjX6o06g4AWoHzn%2BwKDgiCwyTPlA3n2G4p24%2BVPTBlPI5%2FkOIP0hxSPwBbR6V1TlRgo%2FSrh4w0olCAIM%2BNfHhohB1lo8oKQTGLr4umZPa9HAnSllvK0vsMXIEnVAmEgAM2w3a8KCWxpcvGlrZ7vQBcF6wiJ%2FiyF%2Fu9gCctXi%2BLEh9v3jenaqbQCAVz0%2BFqCKqjSWHzp7K52fVgaW6%2FW7182JgHgD%2BZudTC%2Fp60v07fBcrdrsXznRy8nduPop3%2BiDbfJbfG9zJU9ZwRVfCDWPBP%2F7D%2Fx9fUY2oO%2BHwI2jQnTBrXtJViW9wa6IWsuygF%2BNdd7G6CmGGdSfYYwl3A%2B9OyCORVutYJ2VbVkbFZEwdCj1Dn1b3wGKhhVg7aFbRcYtfFaswpUPBbnE8Ipgd%2Fv7yav7riVKgTaM1hy4WlOKa%2F1DopAIZ2IlEAEM%2FtTmiIhGmdCrYsux6c2sdhN4QeAR1OuLOpLLYisM4ARnMiUEAwz31OqJiEMY0K9jU7CjHIfEHrHlU%2FMRdRhzHwXc%2F5dM1cQilUCcOAQz71PCIi0OY0q3gtNDQwHCIDgBwqKMRd6KUtbdP0c8wiaMVf%2FJ4uYJOSBMZAIZxal3ERQbExI%2BeDAysddGh1kXciVLW0O5Z8jN8YMMkAtS%2BaBi%2B6RRBVESg9GtDTwQwniLYJXPSMYG4E6WshN0lg6UBdAygWegWp6ERDUBCA8QDw04DJi1UKRNoQP7ciAagTZSymHUVrzJhYGAUoDOQiQIAQ3aLljOiAAAowMwUClDTHHZrOAtwiAWgzpWyinWZ%2BNHDAmcngk4kEw0ABm1Z8JIgvWsEyPERppvvW9u7ONo1A2zqevibJfGptofWaNf20G7oe2iNWvkentnVsHjnu66GSrvZILkazkTolWeoi47Mnl0NverAbnXgbq6G8IK9gfoHNdj78Tg9a6zPnKahrpIwQgp1Vxi2lAam4kLPoT6tDmxXBzYt1Bvore8b%2FLYL8zyYm0a59tnZvPDyRLldabp4nuhyxbZ%2B6TM681RGlxj4LPbAlnXgLiq1B5ZGKe%2FZkA2CXZmg3PtL9i0JWPI5ZSsphSF2B67r5RudvDJX5hi8j0q3qfBYKpRQtx1d6qlvvt%2BtpqfU7Vo2Yph3sFtTU5QZ9Fzpt%2BlpYiHcW%2B4qYlMNH5u2BIbSXcQmgxu4iSh%2Btjxtw98gb%2BIgDBj5r%2FvA202Rts5IaDS6iaFAbffNaIayszKU0QxTzISndg3N0C2w6yUYU4UbPkQwNEBaLla6%2F5atxOEfOqyXX3QGPvELYJFAnfxYtzFM8RmeytVQKAr4esyyCrv%2BiV5oQLRcBPWFRS%2BZ4gB%2FAwMZ1IlQAMM%2BOQJgJRTG2A5PMZoD9JhjFXoHEJ3QAGdZ5%2FtXuiEeoRTjxCNggV60pxGP6LefUBl5MMVveDYwS4E80IgdoE2UsjxX1jfep376ssbIFHQCmqgAMISTtQAuKmCK2%2FCshThmBBVQ2CZGVEADXmUl7ZotQ46FHzH8ygZoYCYaAAzdDawFiAbAoQHGeA3PZEnr5tYxmwlMiQmgzpWyAvZxveYTMAt%2BDI4IdMUyEQFg4G5wqmxTpwEO4FP9RMS%2F%2B%2FQTed9SIf%2Fl%2FZmMWAJQ77uMqGy7g2SDMBPDisYOQWz69hOqjOtWx0XkgjB19n%2BL7c1P%2FsqNB3E9t8N9McMCQRBu4B5Np%2BZHkYs%2FzOezvXzs2dbxjMxf3LEk5I9pa%2BawXb6d072psXmTSi4OKa26QuIXKeFMNm1V76aJ4dZNnixoi2rkgysBhP4np68QFPU0ze3K7NPU66S8CHUDxJMl5uPIGerOW8dCXq8rEg0zOPFk4ffu9iDy4DQe19U2qin1ziNRDRcg528NiJYVZTydQU1ajwFDn6zCgcVCi7OcB80qOup5qliFKX4mXgsdGYyW12NeVWgDT5RCA5xldVpQimu2foBOKpCBnUgEMPSTDSsqEmGKaYnXQIMeEolQKP0SiTg%2FnEVY7p5I5q%2FZ9ohyYhBKkU4MAhj0yWEVFYMwxqVk3kJVA0MhOgBgrtA0lTiCBrzKapzYaLjyh3Z%2BeWcwEw0Ahm4yQsVFA0zxG5m3kMGMoAEKzU2JBmjAqyyfffzph8v%2F%2BxwFQyMBisxLiQRAwfbZDzTu0H90Mf4wHou2o%2BIE8%2Fmo9xPM6wrn85vU46nmogL1%2FWp5ld7ZkKrlZ2KLVlQXi%2Br5vpuQKuNOquOaVS3vjOXprazQi54Szo1YdHixi7BuHuS5oZZVadJoXkwvpi6gfN8Zk0F284WmEs2miGmqqy%2Fvh1w0kdXVD%2F3c0CI01XAHcr3QgGu5KMOs6nrA0CeTDFixILwCiGac6%2BBQZTTDkEJ7x5IrKwqaoXuHWy%2FNyEOTaAba1CoXbnyOfhLLOAPyiWUACwWy5sa6mWFIJb5jYTTp7jHFkoc37owql4P8K93c5ed5QGcXyKBObAIY9lt0ZhObAMUmTKnKd6wW8twQ6ERXfBCd0AtnWer7ryhMvz1%2B4YnnJUHZ3AcY7cQogMFfVgQ%2FRz85CuJkQwaYqot2JlXDanEMAXYHTEeEDzHTXh0wi3ilSp0Sd%2FUSGnjxrLf%2B%2ByISqX0E64wsEqep4hlg5FO3CbBQIPHsHC2nykiFKXU5NkbFrMe8qlAxI0ahAc6yYlYwiq%2F%2BCuUGF2CsE4cABn6SzFBxCFOqbmyMMlmPeVWhTEYcQgOcZZms4BA%2FNs%2FEIdRinTgEMPC38DImDqGfQxhTayO%2BH5GInTgkEoE1jzqyWHcVrzOznKzkhkiEUqwTiQAGflnXIxIBmUQY4p%2FpOC1UNDAkogMA8kAjloA2Ucrq27%2FSzbfoT39o%2FpndoUwkABi2G3gZg%2FHPFGaWhXXm2B418s7MXDcnoz3%2FTNuajY47aG5fVV0zYTtpKnXDguSkaY0r3oKzamT2ZKUpmRpOqwMb5qXpNBAnwaSDLK4rKcGaNEsJLe10dcR9WcH%2BftyrtKcBFfe2mJXPHfdOZWDD434ylmlAeSRteORIWoRNNyDdc6eus4c3r6l5ri1c44Au9ic17szH0TTkXSYlO7VFMFNLTnk%2F5MqXu8w39%2BoGQ2NOb6lr0tlJgDrF9eJaLoEh89wzQZ%2FaxoHFAlXENKMZylrQldEMQ5p0JjX%2BzXfD9s0topIYBtasWuMHfZPEq2v%2BI4lf9Ap84hfAIsEmfoF0G8OQBp5JG%2BdmMGUz%2FaVYq2spAnELvXCWq2p%2BxBiYBTKYE5MAhvsWp4cTkwDFJExp45m0sW0eApXoig%2BiEnrhLAt8WdUC1kZgwEAnMgEM%2BbIESGRCfQ%2BPOgZhSA%2FPpMamGT6D6JI5PaIImBOlmB8qFOG%2Fty8RcgSdUCYSAAzbskpHJAAyCRATPnYSIHA2FBJgd632JhKgF6%2BygvYljL4lAUs%2BrlDafegEM9EAYOhuYBKttnfvN0viU5v3LsYfKs17M9sbHW3eA918q7SaHlITnifqGkVrktjC6LkHb14Zd1Yd17AWvBpT7EseOEv27fGR4%2F0ooR9RB17XaUxq9Z65DY%2FCEvVFYFk6deCdW3C2qQNv%2F34c7sCzDmJxCAXyNrXg4Z60Zf0tn7SpAa9n4NPqE1gkUAPeuRvwVLEMUxrwasyps6M3OcnQvS2tmWRQFx7q1FrjUk19%2FmeCPtEMYLHQQHLc3eTmU%2FYifoojf3nuDe79zW131Icv3QjO9rfKlSyk7W%2BrjCqx%2Fy0ir3cPusrAs%2BrA3TbAxcAc0f5m51MLUnzS%2FTv8Y6zJ%2Fo9xPffk79x8lFnxWu0oVXPA84zijZ0OT5n%2FXf6gcWf%2FBi2wcCRO68OYP7M9idN1js8DNRLnm40x3rlB4QIE1NxQ1YzONjfYVbGqOrBh4miNBT%2F0hY%2Fach1VKqfAiZiGhf87GfHrRniDtuB2k1Y%2BNTWds06cZXbLeez9Jc94Nm0%2F2b27dAv89WL7rd82T3v04G44w80nhk5w0%2FleurDPNL%2FZlXGt6riIlj5OZU1iT2z1q4XqINZkRouFBnUvqreK2uXcnXXCfup0rHf2i46tE4DtIVli8n43i1q5Cbp5aXReWcpLpu19LRPGlYGt6sCI8qhXWfLY3kR9Hq0OcmFZp48CNpFu35vEcXrkneJbJP7z4kscsOxT%2FwM%3D%3C%2Fdiagram%3E%3Cdiagram%20name%3D%22SuggestedIdeas%22%20id%3D%22Xcqjru9t4SoaPl4brso0%22%3E7V1dd5vIsv01erSX%2BJT0aMn2OJPEcRzfM5Ons4jBFmck8CCcWPn1FwSNJRokEI2oampWZi2DsVrArurdXVW7Btps%2BfZHYL3MP%2Fu2sxiow%2BfAtQfa5UBVlej%2F6MSL9ezsnIiv%2BOb%2BZieH6dlX13ZWOxeGvr8I3Zfdk4%2B%2B5zmP4c45Kwj8X7uXPfkL%2Fmt8e7QWDnf2L9cO58nZsTF8P3%2FjuM9zNpAyTH%2BztNjF6YnV3LL9X1untKuBag5U7c0aaNNBfG73nzYLfD88eBm7ePk2cxbxk2XP1X2%2BfllZ88ePwfe3P6fu7QdbeTtLvsy14E%2FNnlLgeOHJR1eT0X9ai9f0rV0tXxb%2B2nHSJx%2Bu2euMXsJL%2FGNo%2FYhPTVehFYQpyLRoxGkEm9ByPSeITiib48XCelm5m8uTM3N3YX%2By1v5ryD6IHU2f3DfHvk8wFl8bwe1T9GHxYfzhT9GHf0u%2FTPxra%2BE%2Be9HPj9FTi0ecBs4q%2Bi6frFWYXjEPl4v0x%2FQmnSB03nLYPPDUlSpoyz%2F4Pxx%2F6YTBOjpOh9NMMxlwnQ4%2FSZH9690wlHF6br5lFJqZnrRS23vOPrsWGqJrU0CIxY%2FG4Wc%2Fbu5jM57O%2FcD9HaNlkb7dbSxtjn%2B5y4XlRf7BsnOnpr69ZphwF4uZv%2FBjwHm%2B53CYiy%2ByA%2F%2FlwQqenTA98eK7Xrh5msY0%2Bhfd7mx4bgyM6LvOomPl%2FTj6F18ehDPfW4VBhO34M5wIYr%2BcGGbT0H9JP3ThPLHPD9K3F%2F%2F8ww9Df9kYhWpDFKao06qCDjbmdA5zdx9LURc929C1FvfRfGZ5z4sEI5vpzXrHSAGQCl9t9jrz7znvnfzoLT8tNlPW3LVtJ%2FJU019zN3S%2BvViP8UW%2FovlclI%2FSGqJjCw5aTTTUGfV97PeXUXtwaxF5e88KIzfw6tmrWojcwWV2%2F12j2SidgT9cdoxqNscm105XEXZd7%2FlT8pdmDvYGNti%2FDUqdIqOonZhBpdHlswOTmEQlJjGEwCQYK0XOJEbVMXcaj%2BvYLvu8k9MIk2gEYvc55qB8ay3LF%2FFACAQqgBNhAIX4CREGPISh8n4XbMLAdtKJMWQWSIwBqf9UFA7Lfzie7QREGgRinEgDLNDzAS9iDWBZg6JKQhtqhMnA0Ibj377S9PUTMegUrXyAbeoG4fwyukmM3KBDINPsDwvZfLCNZn%2B4sz8DKfbZv0ZoC8zs39qCKrFBIgdYXSgfM5vFrvExvPUxkgO4OCfuAAv4fISNuANc7sB4AnbuUCPKBYY7NHCaYyIHiH2kOuTQerW03MWFbaOMKnQIZJr9YSGbD5bR7A929s8q3pDP%2FgxkW6i7hlTqMDz1okptWo9F%2FKBTPPNxsGlgeY9zXKUO%2BGBPbAKWHfARNmITcNmELgmbqBH9kmAvITEy4gpYfSQfNbuLQBkNd%2F26%2BevQxVnf0CGiiQbAgniN%2BjOiAZ3TgJEkNKBGIKsPWwojogmYfSgfIPvT%2F0H7Ce1inogEKCNgd0hEAgOR0JjOHXIiodWIiUmwn5C8DSIKWH0kH0uLCxpu3MCxoZMFYECm2b8rZM%2Fun25uvPuL%2Beu1aVln17%2FvP09JOQnV5I9PhLEQdBilkyq9%2FHITo6kfqYPkN7w%2BW5717ATwtwlgwZgmfliUtiydZv%2F8n5vrSbv7ANpz2t1nynBcbQrPEge6n8OL8UNpKNWYY0Oh2qZKtXLJd2t8GkrP9bspUQX1LMyvvzEmtZ5ev5uyWeSyA8pmOcU%2BlCg2IYmEtwYum6VLcQ2NkllQu1A%2BmSWhEqTlLRroRB5AIZ%2FtlBB5QEEeJJHz1jFmsLTmUxMbJPKA1YXyCS7fwsBxwlh9w1mtiD%2BIxDrxB1jgp0akmPiDLMLeeo0IWg8IBHUiRe1DCwJzfuQQI0DatPsgFOjEHmAhn1JoUbEHSYTBdWmTaPdYGdEDrE6Sj7TN3HCNkRh0iGGa%2BWGBmrqIopr5JZH1ZsbWl5mfOoVidpIGHyb7FvauGVhjENPUDwvVfLyMpn64U78smt5Gv3qBJlZGUz9WJ8kHuFC3%2B%2BoQyDT9w0J2hcRxx352WH1MNJXO%2FWffsxZX72eTCppk7lfUwU54yfHsiyDYwO%2FqPgL4g%2F%2FZ8taMMLz%2Faml59pcN%2FoP42TqMNDhvbvh3%2FPP5cKikx983xyPdTI8v37YuvmTcIgJysP57%2ByD5O4Mdvv%2FZ5oj9nW2t5pvh2XB3TuBGb2tTFfzOAuKnUs920utW%2Fmvw6OzjCqm3CRnnKbtQZB8IhclxbeM%2FOxk4Cyt0f%2B7esDibvIsJ3btzUM00JJCXCmMjJo8w%2FavaNlg6rpYbV8%2Fbf%2FJGmo4bgd5ab31qSmePenp77sXYuZczwzz6K5cPwh7QKQfR09LZ4x5%2B9HfJ80fttfn92jtrvXRSWJLmgSCSkpc8qFotBl3ygN0HLfpblTwwmu6VySV5YPIbqP2WPEjMkHYFkM7CJr91Gs3C984T%2FE2BTiUPGsOe9hBg2QFVHZwihCCKTUgieWDye7LXH5VS4PWgh4dJhQeo3ShfeHC1fFn4a8fBpaGED%2FhEKGBZAhUiYCIUksggmHwdwvVHBFkJ7blVqlRA7UX50O6dtX5YvyCgE7hwTvQBFvCpmgETfZBFBYE1Rt3hD1qvNySo4AGzHx0VFDxYC%2BdLYMMnELhxT4QCliHwgT4iFIAJhSTCCKN%2B1UgkVkZsAauT5ONxd5ZrX%2FauQrIxjmn2hwVsPi5Hsz%2Fg2V8ScYRRjSCYDLO%2FQbM%2FZifJB8%2Fi2f9iWV6RIOfk3xTGNPnDwnWNxjo0%2BXc%2B%2BcsijzCqEcKSYfIf0%2BSP2EkyA%2BPLEWOBpFeUPRc6BDNRAFjortAep6oAwm8n8OsqIHB6BJmmQSaFYAz2CiEc1G94l0o4n0wmgx25BE3RBvsFEzZHJxdJSK47qJEgMikelEbCcLQ7QkuaCIpaVgSNUBQhq8tmdNE4Xkqg8iBnaX6s2EEUU9gg4EUR%2FB%2F%2Fi2ak6OeF9cNZ7BCMOJ0xeRBpbuOmR%2BSWk97v2o%2Faqk8nDRJZONqX8WTnTNVyeiWZXz20lMvyx8T42vdvuU2DMkWIvDccTsb811QLvqZ4otTEDI83xugvUnsUTLOoirTaTkszqz3O5UmrSTEuiFr3WpNiTEWkMElQRTwXdK%2FCUvTRqShFY9zTrg0sQ6Aa0lMEbkTRCUlEKcYYe1m151OpgBS1Cy0tIN1ssgBnE7iQTuyhK%2Bg%2F%2F6PrN%2B7w6eZf44%2Fpz993s4uJcqZS2gcm9oBPgaIYdRizPkT41D02SOwBqQtld8izh0tn9dhX9tAO0ok9wCLOFfx4iy0zChNGdntnHEgYObJzRnKT7eV8jCsnfQhceEJK%2BuD6LSiqtjtkS1kgej6nQdVz6SfNskCgWTAjlLz0gRPsXwgMKPpfycLS4UxjN%2BwfIa0alVeZ3Er3XL4YQRWyBmkF2RiGiaFSOJs9DpUn3b0OZ08EtsiiReTp8cwnBaGUIDp5OLsx7mlJCcsQ%2BDwlohPiN6RF0QlJwtmTGtIXUm1I77FB4hJYXSifEZRyCawaRXCRTuwBFvRrpCURe%2BicPeALZxc%2FjhpJFD1gDyNiD5hdKB%2FTS9lDrHFA7EEk0ok9gIJ%2BFuwk%2BoCCPsjSUEEZ1oihyU8gUjMkBoHWjfLBuWlgeY9z%2BLEMZEAnAgEM%2BVTbj4pASNJAQRnWiJr1oB9TaofEIND6UT4et0lJnM2tCKkL6CwCOfiJVQCzBirxR8UqJGnMoAyLGkUrpdDrBbGgSn%2FcrpQP1c2s1dx1Avh7E8iRT6wCmClQ92hMrEKWjg8Ku48dVtHz7QrqH43alSp8%2FC5OtXxwl87MX74snNCxiV20agHELroyidn9082Nd38xf702Levs%2Bvf95%2BkZxUFQcQtW9Y6HWxSijo%2BCXH%2FUesIsys2QeAVSJ8oHQWavq5A2KlqFPBEJYNyaTybi8H%2B48VObfauOVRE6RrFoR%2FIoeQ7dt54SuisOSYbIyKkBafmWUC2pEJn59k2IW1GZSu5e2mhFlR9E1yYNngvwLlFVHSe%2FBrt0FpHtBOtNLsUH78nfvyjLLcBI4qnEe5VY7ZliTDgSoRa5Nw3OuqoES6TKUG093xCISlP2Wq7yVAw86Lgr6PLyMXpxsNIQTl8hoQhUaOh0VdPXRU2pRAOu1f3J5Z6aI%2F9tF3i0vu%2FYFNQhUYsThAqEUYsCxSeU1ELlY7Yde95Oay4TMyRGgdaN8tuk0Wp%2FFnvIx%2FC2fKEPhFQgAzuRCGDop3wDVCSiQPgJJ4kAV3fZLYkQmG9AJKIDOPPbbRGJ%2BBYGjhNe2HbgrFZEJIQCnogEMAugUktURKJIAgonk8DYULmJ5xSYMkJUoQO88oWUEVW481eRb7EdjCyhSzATDQCGbqqNxEUDCiwHJQ0oaCYrNw0QWPlINKADvBZUPjqLmRuu%2B0YBBBUwEgWAgmw%2BoEYUADIFKFBdwkkBasSyZKAAiaERBUDrKPkY2CZogLTXVJdIJg4ADNo12gISB%2BieAxRpJOHkAHwYqnuNpJZpgEE0ALWv5ANYablhLIFkeWv4lQ3QEE10ABjEa%2FT5IzoAgA4UyBrhpAM1wlFScIExcQHMjpLZ3RZeH%2BZuYN9F2EzEB3BWJHQJaSIDwDBeo36MyAAAMjCShAzoNQJTMpABvak8C5GBbvHKx7MitC2iZ%2BTY076lCTQGM9EAYOgmZSRUNEBjUz56GsDHproXO26ZCZDqEW5fyUe1bqxYSPVLBIf%2BUQFSMpIM3hVKuEDrEJ8PlV0t4uF4MtinRhwf3DmBG72AjSDohm2cUqE4fcCHFYpFlpuBUihmdpXf5WhboTg3rpIfF5FCsTHZvRc1tVuhCsX5QcwGT0UOfWK9sIJwo0%2BcZgzsX0jlFk2kTlziudhww5w8saHwvKGo50s2nYNdCVGxXrUleEMc6k1DWOXixMCbDRU%2FD2YXOXHiThcx3coS69THEPWUbPAxzdyUDD%2BJr1t54sYWQO2HgJlEjWhrr8lFw%2F19UeSiQJ4YJ7nAWAXYnl9NzJCYBVo3WlgkuM0sbq0lynpBwJgnLgHMCKicEBWXKFApxskl%2BLhrr7mEwEpD4hIdwJmPs7I%2BB69LohBCoU4UAhj2qQQRFYUo0ifGySEwliC26FgFVigShzg9nNk24Racr5aWu6AuB22AnVgEMPRXqF2smqkY4bt2omKacMh%2B3kpTLE81ZMmNyqBWcuNJ0xGzVJTDCYki5W4gJSRyaUbq8JwpGrackzg%2BPHKzrERwdjziSdnsdRXNKU5w7Tj2D%2Bvxn%2F2rgwHl0FUyNoawXBLnmVF9i3DPnAaB37MvTQvLdpPoRk0TRCRLohvzVKTnSXTJW6SlJdZJeaxyiGaTMWXPtQx9WmgCswXqy3uS7WpRrEKW7Lkx9eXlzZAoBVo3WqD1kFKKS6Qy%2B4DBTiQCGPprpC8RieieRMiSNjfm84yuPyql0OsDjxCookA8ogNE89lDcbwA%2FrYEMpgTgwCGe1IIQMUgpMmaY1SI9iG27JD4A1Y%2FOuGDdde%2Bb99bYTRdE4cQCnXiEMCwz4f1iENA5hBFupgoOQRGJYAGAEgMjUgCWkfJx96uvJ9u4HvL6M3j5QpdQprIADCMUyU%2FLjLAJn70ZABjKX8Tz0m1%2BrgdJR9D%2B%2BYEP91Hp59EgOrxJcM31eOjIgKqIgsRwFiP38RzUsE9ZkeZtdjarvMJeksDqKBeMnRTM2BcNICViiOnAeqwRlRKAhqQGhrRALSOsqgZ8DIODPSLAjQHMlEAYMimRsC4KADrWIeeAhQ1ApacBQhsBEwsoAPI8lGsaWB5j3OclQhdIploADBoQ2kC3EQnL%2BsIPKgu0JfcZYvaekOBReyQFPMmjHSkxETPW0lLcnn5cdXcsIg6%2BHKPcDhp8JWlaK%2BrDitEJiF2Iy90RMNz3RxtO6O9rqioLXnb7okpTR5W%2FhSp7wXJj%2Bk5%2Bc2zLNbadi9yMz%2BwmRsYkSvTjfzNpI%2B1176sQrC7ni9LPFZFV1bskmpwo2N9YeyumBqbH4Rz%2F9n3rMXV%2B9nW3Rqr%2BDro1kyBpX2QvNooZ4%2BjfDf0lpxaVgLLurVPcuMi8mlm7l7U8fH0rPIghnm83%2BQHafNDz1Sj9x6eUYXtPFFr4XwJbCf4EDrlbZ0QylgXFZ0Ojt8Qb%2Bh3M912tuxkAsOsW%2BGh%2FfEslg51g1yh5IzqkRkh1c%2BpOTcP0kiich3N5hwEY5XrWfd6UlW0rlvzXalpimGNZk1cCN3urjK4hLvdCp%2FOkU7b8CM5VaSuASP%2FbRd42x5S7dISKo0uoylQ%2Fkc1liGsqYswliGJ6rWq8JkgKHpptOhlBSaKEL%2FoANF8osi3L%2FFC%2FNYnetEq8IleALOECnklRC8GEDcxJNHDjr40B0EUiaYtetkR0QvUTpWPK392vNc44AB%2F%2FwIZ1IlQwMI%2Bu20iFOgIhSzy2NmjQ1W92p6PVZvig%2BhEt3Dmo3xfwzXxCKEYJx4BDPTU4vMUda%2FiyIMkutiqirHHZxPPKbD%2BidhBB3jlo3NZeuO30ApfVxiZQpeAJioADOF8tI6oAGQqIIkqtqrWCI5JQQVMogKoHSUfSbt0Fm6EhQcffmYDNDATDQCGbuq2iYoGyKKJnX2%2FnQQFTW4mILDslphAB5DlI2AXq1U0ATv2Q%2B%2BIgKCWmUQEoICbj4dxkK4q2xABuLYCDUAVq4nAgjdIMgljLd3dZApMJ9J%2B4cqW9WFu4GY6CfBsqkK4rU1Zp10tlGN1TZhlng%2BH6o51jpTJXvssknbqXiWlqvaTULoGyfyNnACbllcraUvDLqc5ZeTVWRCppGQKL2xZZqpHf%2BXKg7AX12ftEY0PJMdd7eNM4FIGjlB25HhmLkpiRJ3skoTKEiPZSbA7D9QPttqWV8MMWq1pWZtkuiIaH3FFUfFblFQoJsdao4axuCdjPpSLpySnSs0vYOiTljwwW6DmsicJpIliFbLoiGgYe8u26Fep9SxqN8r2iAooxaWzeoROKpCBnUgEMPST%2BikqEiGLWoheIfjbIxKhU%2BNa3G6Uj7tOrZWz6WFPDEIo0olBAIM%2BKZuiYhDSyIPoNaJqYChEE89JbW1xO0o%2BGsc2GmZW3xrcNwYz0QBg6CYFUlw0QBahD71GGEwKGiBQVZRoQAd45cNnFz8td%2FHfD57dNxIgSDWUSAAQbLPb7rikh%2F1cq%2FlqVrFwRCPq5CbbqyDIwjSHawhEik5DqiFQFDOXHjw2dodsrdVqbmA1P7BkNUQGHxKM3Jz7Ur6RjTDdHWaXTSNXJ6elRR6HU%2BDzlS3QWLpRI97X%2B%2BWhmEiL0RSVkmXDG3yMjrpsZqYphjGQSEQHuOYDcHLlxAOGPmlKALMFqrSrXmkHi2bIkh5vFBfdzbpvhtUxzRBYekc0owNc8wG%2BD95PYhknQD6xDGCmQJV3WDczZMmfNzAW4bXoYgUW4RG5OD2c2W3v9sa6S9pfQGcXyKBObAIY9qkEDyubkCaXnsUZiU5smSTRCbQulQ%2F1%2FZ%2Fnhl%2BePkeO5zVAWZIHGO3EKIDBn48IfvCeg8ilOl55oQnCLJ7jQS8qY2eUU0zVqm4wgBetNCmUVj2U1sT7kmjl7vMojp%2BBj5y1VzJvkmgl7ulY8sgZYORTgQgwU6DI2UmqREWRClmSckyM4bIW%2FSppVqJ2o6NhGaO4tZYod7cAY504BDDwU7wMFYeQJeVmhDFG1p5fHZFkJW43ysfIUg7xsN5T69tLDtEY68QhgIGfRCtRcQhpEm1GGEUrW3SspGmJ24%2FywbqZv4p78Mb5NkQihGKdSAQw8JPkJS4SIYvk5ahnkpcjkrzE7Sj56NvXcP3Fu7F6J3nZGMpEAmBhm80f%2B3xxLAzJkmgifLjh%2Bn6jneh7V%2B%2B%2FmZ5CGPN8Mhlsa2OONHNQSR0zO6imjhl%2FSLQIcKOXtEkbfmcALYpkMvG8wyKZIiWvQIlk6jkBQUM%2FkUhmXrlQzw8smUjmuEIQUhKl29bt1qxutwIzYUHZ7YhNI8x88jNnW3Y7UnMDq5LbLR%2FEzbq9unu6vSIsjoEpcZvtgLMMQzZxHFqUR7ZZboUQVuVjPqC6H0593g4Ss6U6bsrjJKudGfNhzVR7rhSJcApo2itwTSxTDG2gcu4OYM0HTUnh9kTQp9puYLbAR1yJZbSrcCuKZchSTDPm454oKnRb9LImEQzUTpUPqF4H%2FvIyukmiF60Cn%2BgFMEuoUSjZa3oBbxNDljobdh%2BosltadLET4haYPeqEjxQ%2B%2BBiYBTKYE5MAhns%2B0kZMAgeTkKbaZlIjJNcDKjFpig%2BiEt3CmQ%2FvxUkLWOt1AQOdyAQw5JMY6UlKbYQxCFlKbSY1wm1gGEQTz0ld%2BnA7Sj42F1OE%2F2wOEXKELqFMJAAYtkk8FBcJYBM%2BehKAUT20ieekbnqYHaXGili28yVd70tgO8HFEqUqR5dgJhoADN0CS%2B9%2BO4EvpvZOrVh8lx1ULKTdKh2OuMHcf%2FY9a7FdNpzce5tltemu4%2BHyPJF59qDK87JyPFZWywSY2y7P0yYl1VJiyvPYwBHorfXWp6as9ajnV34zKtsUYrWG6ds85jtXH8UcHz%2FKxoslDwe1y1QqxGLruczEMdb0mGfD8%2BFwvOU24xPmaL%2FjLFASsK3VfPPhSgPP2rbf1Cp6TV1g7jAkp6np6rmqTLL%2FxjtWGdnpuTp6%2F%2B3oNDoFmp5zDjk%2Fjsid5m9FVVvwpvlBFP0EgxgjctgVEhcOU8MTuHQl8ujKDhHWlANMuMChQ5WmYLl7B524oggUMoXkxSc5NsVV4bfFfIe5gZX8wIhc9djcvRdWpy3Ui%2BYHYZjsiR9dWR9vnv9VLOfz8814tX79c%2FLn09lQNO09bqeA473Dc2VygPa2tl8ARprr09crxVP%2BUr7fn339%2Bl%2FTGp79%2FdRYZgSoG1W0sX6uTnYMNKM9bbvSzGcz9zPJDYzIlSpqzs1p6RwteBMh77EVo0e%2B9PvPa%2FUf%2FZP%2Bl%2Bndvpizjw93t%2F9C8aU84VQPaRgKJJwQtmOrulJJGamiqzkPwJhi6xqH%2BRU3Zj9qDHdvRkvrNcT60fwoqt4nP1polrwfZVo2nD8l8TnR4nO5cIrC4jiHMlSyWGv3KSrltJkyo1qptSq34uYpUviE5wqfBh9eSmXnlFIYyiUL0%2FZalrKoTo5pfgd%2BGt3ZwiHFuVZBT9lWoKyA%2BiRWIxbHqM21SSzwac0VPg2%2B7C8lFuUw7AGxELizQ8Ti5JjmKwpJzPYksCdqAcoOqJpLcDVXm3wCn7hc4dOQtpCr3MKIJyD1jyxqv9s1ETpDgIVgmvK7grT283L8ZXnhfjVuP%2F%2Bc%2Fb18NYzXKjkEorNaa2YZHJl2tVORkNzT0SkAh5NNdUlD%2B0Y%2B2VQf7Y7YUmSfC%2Bwz60QY2De5iPvx2fSVB1HSrhT9COurX9UfirX%2B9fwhGD1Yv5cXmn5W4Npmr6toDnWC%2FauaAaqwfqWZvM0Q%2FiSHvDOl6k5ndrL7pUkhgCiEX32nvRIMy820j3H7wqdRHLcHv7FetK1Y10mV26AYXqXVRIPQ5UWVwVGvLgrfHh%2Bxj6dgOTbVwUL%2BbVDqEBmb72aFXWV0%2BWyA4vWCN9XbZBH4gvSFT6OGNi%2BYTfXW%2FKnArRmiECeHMh%2Bbv7WWDnQCgQrgRBhAIZ7XCybCAJYw4IvCFz6NGpkf8hOGEREGxO6TTyj5w%2FHsPZv%2BfaQMTSFOlAEU5gsSU4gzgOUMCJu5FaOuRoBMftagNEUH0YZOwcxH3maxa3wMb32iDiJxTtwBFvBrdOQk7tA5d8DXxq0YdTXCYmC4QwOnKVCHn8jB6dHKB9Sulpa7uLBtlPsKHQKZZn9YyObjaxygT5LeX1P2b7fhAOv%2Fcs6Err%2BzPyrVIEzuUogqYCvzPdCCgMl4cm6kEMkSfvOqey0VBRSNLVjxr33z3Fwb%2BH6450r2LaJV2%2Fyzbzvxp%2F4%2F%3C%2Fdiagram%3E%3Cdiagram%20name%3D%22Official%20ERD%22%20id%3D%22YHijvIm-SzFwgbNPgBDP%22%3E7Z1rc5vI0sc%2FjV86xUUg6WXkW3ISO17bz9nsvtkiZmzxHAl8EL4on%2F6AYGSJQTKIAbqHrtqtMgQxAv275zfT0z1H5sn87SJ0nqaXgctmR4b2GHrukXl6ZBh6%2FH984sl5ZFsnkituvd%2F8pJadffZctti6MAqCWeQ9bZ%2B8D3yf3Udb55wwDF63L3sIZuLXuL13Zkw4%2B6fnRtP07MjS3s9%2FYd7jlDeka9m%2FzB1%2BcXZiMXXc4HXjlHl2ZNhHhvnmHJmTo%2BTc9n%2FmSRgE0YeX8YvnbydslrxZ%2Fl7%2FMa%2FvjH%2F7d7f%2F%2FHny1%2BWd9rc2fDlOv8y55Luu31LI%2FKj11o209Rdn9pz9amfzp1mwZCx789GS%2F5zxj%2FCU%2FBk5v5JTk0XkhFEmMjNucRLLJnI8n4XxCX11PJs5TwtvdXl6ZurN3O%2FOMniO%2BI340eTBe2PuTaqx5NpYbt%2FjmyWHyc0f4pvfZl8m%2BWdn5j368d%2F38VtLWpyEbBF%2Fl%2B%2FOIsqumEbzWfZn9pAsjNhbTpsfvHW9jNryL%2F6CBXMWhcv4%2BC0n%2BmV6POQm%2BfpuGPooOzfdMAqTn3Qy23tc37uSGuJrM0HI1Y8p6Ge%2Fbm4SM55Mg9D7nahllv26m1paHb9685njx%2F7BcXOnJoG75JrwZrOTYBYkgvMDnwmaSy5yw%2BDpzgkfWZSdeAo8P1q9TWsS%2Fxc%2F7on2yTqy4u96Eh%2Fr78fxf8nlYXQS%2BIsojLWd3IPFEntlicwmUfCU3XTGHvj9w%2BzXS%2F7%2BFURRMK%2BtQqOmCjPVmSVFZ8HW3EDQ3PW3naqL323kObObuD9z%2FMdZqpFV9%2Ba8a6RASIU%2F7frnzP%2FOee8UxL%2Fyw2zVZU0912Wxp5q8Tr2I3T4598lFr3F%2FLstHmTXVsSEHs6IaqrT63vb7j1G5cWcWe3vfiWI38Oy7i0qK3NLl%2Bvm7VrO1swf%2Betqxqnkfm147WcTa9fzH7%2Bkn7ZzsLWyyf9vtFLnqOjGDUq2rZwc2kUQpktAgkASnhg%2FxFTZJDMtrrh2Py1yP3691jLCbxYh9UmgcI%2FKNq%2Bc%2BR4KUr5z57kE8EIBAJfCKwNCa4Eu1rp7ixwQMeIBBLz3hBZsY%2BGMQMqxNkJABqQPVdUHLF8x3WUjUIFHjRA2wRC9GvAgb4GLDQBFsqBAnA4MNh%2F%2F6et2fn8CgU7WKEbaJF0bT0%2FghMbJBh0Km3h%2BWssVoG%2FX%2BcHv%2FoSK9f4XYFpjev7EBVWqDBAdYXagYNDtJXON9dBVghAO4Oid2gCV8McRG7ACWHQy%2BmBw7O1QIc4FhhxpOc0RwgNhHGpqg1rO5480%2Buy7KqEKHQqbeH5ayxWAZ9f5we%2F%2ByqQ7Ae38usg3VnUPKddDaHlQZdROyiA861bMYB5uEjn8%2FxZXrgE%2F2RBOw7ECMsBFNwKUJWxGaqBD9UmAuITUyYgWsPlKMmv0r%2BHW3fOrZGoTaMqa%2BH5auK2SdUd%2Ffed8%2FVqTvrxC96sM8wpDYALMPFaNiMRvQJEKzmieQAGUE%2FAkJJDCAhGmoARJmhUCYApMI6a9BoIDVR4oBtCSL4YsXMhc6LAATMvX%2BsJRNpRcx9f5cpNh7%2FwqBKxV6%2F4ZrK1Lv36xaxYDXpeM7jyyEP1UATMjU%2B3el7In59%2BjL6S97%2FHhz9eP55dq7%2FLYsqNst6Jm5j4zPNcUd6TR4DHxndvZ%2BNp2NSnt%2B3dieZWK%2B%2Bzmp%2BB4fnt38ZmFwF8SWs%2BS8kP%2B3HysLCJO3y9bUsAEWo%2FSWm0dvXvQzaeyTpuvZ8V%2FJJz8NbSs7Pn3jPj45WG4cXLPQi3%2BIVanx1Tk%2F%2FlF%2Brr97fJDey%2BKH77daHfF7pS8teVMHUEH8toPn8J7tM7v0uohDUBv9zLqM%2FqZFrE%2BGbOZE3sv288qz0uuE8N7dhWFnE7U8mmLkLDN9g9mnKlvlznbNXLt6vt30F6nbbmwGznLjrhnfHvT2dr9D%2Ft2zZ7FHB3%2Fj0m2YtlXjtcSfS98MEg9bbJLi8oN0qeL%2BUVbO8dHGCDscFhceHwNlwhuUHSdBL1JvUpS%2F3PC8ZhFwW9L4vKTsBsBVJ0b5e743QsNx%2Fn16aHwAn28c9TCn%2BOcT4%2FwY8wXa3xuh5VB%2Fa2ZQqnX17IA%2FIdFEo5P9smhCkbpFA4yh%2FsZ8amqDFAvA6kLFKdMUJWifBNlCp1gBLOXTSgFM8KArUrlogHGpQHNOlVYSoPah4kqC2yhkLEoqG7HFggBCptYJIGCJXwyFEUAABghFih8NaHtGwQgJILD60ILIXBA7xFiQLk0%2FSBU60QMs5dNGjajoQZFiR9zYUNFDDa9JezFidpKWGGo78aIlRjDoUMPU88MStRhyo54fcM%2BvSKkjq1%2BbLaZWRj0%2FVicphsluo95ttFhbxNT1w1I1bbSIqetf539h7%2ForRKpU6PppK0XUTlKxrRQ7FDJ1%2F7CULc75XzvLOcuyrSl7Vo7cR1xdPDW8ZC%2B%2BTlAA2o3z5yB6bDR51qo731oxeRY4PNriHHy%2Fk2dTMyS8RNoJ2%2BL0e9wJ37AH%2BHTZafJsbdkTjMKyA8p%2FaWMuShZNKLL%2BxBbn9c%2B%2F6YBwQmvdr1IKDGo3Kk7pxziR7MeFqxgHPt0TT8AyBEqHwcQTiqxqscVgwfk3BNGt5twqJcSg9qJiQszZ%2FGkWLBkCnsAldOIHWMqnhBhM%2FLCuHY0cIHjMbwsgzF5PSFDODGY%2FOhTjdbfOjP0IXfgAgVv3BBSwDEEM9BFQAAYKRfYCG%2FYrzya1MqIFrE5SjMddO5572rtUm9o6pt4flrAp1QZV7897euy9f79SbYaUaoPaSYrRs6T3%2FzzfnZCgZudPiTZq6VqMolHnD7fzN3RFOv8KISwVOv8Rdf6InSQ3MDEbMam08YyyeHeHYiYE6Erddy%2F%2FXM3ZxP766mnPZ4%2BTt8e7h2OdlhOgQgC%2BlxseBCiUnSE6VUUQYI%2BVEQIgdZJGQW5vigAn0%2FglqRsCaEbMhAAtqTv49f%2Fxk8Z%2Fz5xfbLYl3CTzJn2wLA1ntZHdhiPez8QHRYEyU6PyHQcLVzQiXeeTAhUTFvh1cqyq2La4ofOcl%2BwrHuvaeCR%2BS6MV66tjg4dbYvyJzBglD04pO7kcvtcz2cP8nbK1TkYFqyF6XetkRMnJMAmopJ4VSU5uvdhJbd3TUACWIVBychuzgbJwQpFiJyPaqk%2BwQWIJrC60YKu%2BjRkW4DSBS%2BlED6CkPzY%2B9uMrXE3BQDe2hcR893MYrrR3dhOr%2By64dPwlp4n3f5o7vvtjJf4weXOME4XrLKarg9XN3rzoJ%2Ffm8d9%2FJX9%2FsrKj07eNfzrl%2FBGrNVz%2BXH%2Bb%2BGDjU8nh%2B8dWR%2Fxz6UMy95FV0z%2Bv5xU8h%2FfsQ8o2tIhjz64LZdbI0rUCDa9PhmzmRN7L9gPLs6vrhOk2DJzbznoC0TC3m0zfYfaxg%2BcEhYYHg1zDBt9WmDec%2FiaHNgzOgsXpxCwploX7BwJHNHlfysJ21N42BwUT4oUorxugWX4sTg3SCFL%2BhPS4rqevOCE9sGCrrmACr9cT0qkZNjeI3KeHxgeR%2BcYVJGlxHg5lcYrWJ6Rr677ikLI1OyjVuoKGUGFqsNc4UW9CWhZO8Gxi5BPS4wp5UepPSKc2SBPSWF2ouMI%2FYwms1SvgKp0mpEFJfz1dSfiAAh90TQ1%2B0DUxYaDHAJGZIREEWjcqBvYyhEhSYAkhpGqdGAKY%2BGmJPSqGMFVhCIhbgHWLEbTKHrcnFaN0k9Dx76fwgxrIhE4MAUz5tK4eFUPYqjAEuJX1nW5dkNkhEQRaPyoG5lZrE5MyHD6bQacI5OInqgBmDVS7CxVVjBWhCl5xBNRmoh37VirwhdqV6mLA7uR5EcGfmEAue0IKYHYgRvoIKeAihWGURApjuNtwQCAFuN3AunWsqR02xxP79NA4T%2BQbV9GPisG7ZLXlnTdnJ8H8acYi5hJaNGoBFdGiNYso1bqKJkE7jaFCC8tQZLZCjL31fOfyzBRptgKtKxXDeifOYuqxkCYsGlY%2BTViAqsxfMBVNUAEYKkbooKJYduDyO9pyq3uskHgCqxMVJ30vQsd374LEQRJPNCh8wglgZC0uqThlMy%2FWxnK10Oir%2FxDsB4wcTFAhtP3Kt8zsjWeEMCowBaOoFKEJBxGK34%2BhEZqWQtO6QzpJW%2BuVFh503RVsVvYt%2FuG6Tx7qtCJaZo9yILXTvrmnXTN3n1QTrXXlv20Ljyi1a1OglOQ2Zr2koUXBPh040aJCcd8e5CMbEvORiSg6kLO4IiEe7J8kHvI%2Buto9zgcCFcjEThABTP2Uk4wKIgo29sUJEeBykruFCIkpyQQRHchZTEmOIeI2ChmLPrtuyBYLAgmpgieQAGYBlIaMCiTWkSnsJMG%2FHyqSqOM5JSYZEyp0oFcxMhejwnWwiH2LyzBSQpdiJgwApm4xPkcYABkDCiwHJwaAyx1uFgNSQyMMQOsoC3KD2ezEi5Y9Q4DaQiYEAKZsSvHFhQA89QY9AlSIZSmBABITeAkBOtCrGANbBQ2QbsjWpZKJAYBJu8LemcQA3TOAwVOo0TOAGIbqvippwxgwIgzA7CsHWhEGJNmGSZEwx1%2FCz2yApmjCAWASr1ApgXAAAA5wn4QdBwYVwlEqsMCgbkI2sUC3ehWjWHdTL3SvY22mtQdwZiR0KWmCAWAar5A%2FRjAAAAaGqsBAhcCUEjAwIBhA7SjFeFastln8jpg76dsygdpiJgwApu4KGWCEAd1jgMm7fPQYIMamui8F3jAJDIkEUPtKMar1xfFd5v6I5dA%2FFKirZkIBWPK2ClNiVvU2sxDYfjLIUQBV29yv%2B3UlIr4AUCuI%2FhcW5F4LFGrfzrcrIahsttymVVeKu8ttAq8Ev%2BN9iPP2SbnNTrvlbgttppYoBzmpGnwHii7MT9nsk%2BEvS%2Bm24GZtC6Cy8MBMghJbWpmxkgUXBQU3ccIFxryWBv0q7VuH240Wpr1sksWVM0eZAQNY88QSwIyAEmRQsURB3U2cLFGhTFsfWEJi7gyxRPty5oi%2FtUQmrdz9PCeEkCp1Qghg2qekGlQIUVRxEyVD2BWCbD1gCFtizg0xRAdyFmN3Z3PHm1Hd7ibEThQBTP0lsnFWcbQUEHRjW0rMdz%2BH4Up9Zzexvu%2BCy2S5TkYV7%2F80d3z3x0r%2BYfLqGCcL9uZFP%2Fm94r%2F%2FSs5%2FsrKj07eNy045erjOYrq6QfoF4tf4k1%2BWHGzcIjl8v8fqiN8kfUjmPrKKFpC9neA5vGd7sSF7jxHnn51XyizgoBetyF2fDNnMibyX7UeWZ1rXCd2927iu8TRinkmkfeJlOnmj6XvMPljZmnY2PTa2W9Z5DhNvN%2F1VDm0XnhWLocmT50Xco7DwnDH3l3P%2Fn%2F1jgyNaQlfK1PjSUb5kkeur7Ao68BtW2xWCjb0eVdYFqboRHcVW0NliVLDnK%2BhSS6RxJdo%2BuSDPKOuLaelcw9KnUSYwW6DdoVqZq5ZFFaosnePDPpqq3jBDQgqsbnQohvw4UpwirRoNWOwEEcDUXyHuSBDRPUSosmZuKAYIz7%2FpO6XXA45ILZE4Aq0nFYN%2Bk9Dx76fwJyaQCZ0YApjyKYcPFUMos2huSEl8oh0SQaD1o2K47jwI3BsnirtrYgipUieGAKZ9yt3DxRBFhd5QMgTG5L06npOy81A7Sm5jmyvr%2FRcvDPx5%2FMvjZYUuJU0wAEzjlIWHCwYKdrtGCQOjCtEwFWBgRGl2uB2lGEW7ZeGLd896CQK15UwgAEzftK0VKhAo2vIaJwj0bFurkcRtrQgEOtCrGAm7DnuLAZK2tSIMgKJu2tYKFwYUbHWNEwMqRKWUwACJe1oRBnSgVzGYdcNePPbaNwKQtJsVEUDrwv7%2Bn2P3afpy%2BzIYRa%2Fu%2Fbdrb6IdUzwAV%2F9fsLs18P6%2FUHZiNOD8myoAsNvOqPtH6iULggHOjP0IXZwZCN3JmHp%2FWFjLk96o%2F8fR%2Fxdtaw28%2F98hvILccMW3tU5tjRgAra8UoTWpWqgwAjSkY4IAYMKWWFT3NwuDylV1DyyKu11Zl5fm%2FaRp5tFmeV5jPD7aV6A3qarLi4fFrDANHgPfmZ29n11dfs1CL%2F69VuVF32GguYq8ZhaZ%2Bbggr8z0dkgFeQe5erzH6zUQDZfjte18w3auYcXq8Y7FGHc2vmXh14jt3poGYTHeosy5Gp2hnK5Q1%2FXxluRMvnvrx5y%2Fp3cEAfoUYS4%2FwpSTwpmas4zBpiJjTTHWnBTrPem%2BLE6Zkr3N%2Ba6xxJi0XVEXUsG9TOMqcrsYk0YzLV2mYi9g5b9tC2%2FTQxpdWkKp1tUzBYNTEFFGiX0AYFGGIsV7DU2c0UaxJUBjXjYzSuILtE5VnPC%2B%2FZEMxK8CwotGhU94AcwSxAUghBc4JjEUKetraGKQBsWSuQa9rEl4gdqpipmgl8x%2Fhj93gUzmBBPAdC9G2wgmcMCEKvV9Da1CVK4PKFFXH4QS3cpZjPD9ES2JI6RqnDgCmOhpl8JW1u5LgwdFCvuul%2BGhgoc6nlPiPoREBx3oVYzMrZc23kZO9LzASApdCppQAJjCxUgdoQBkFFCkrK%2BhVwiMqYACqaERCqB1lGIU7ZTNvFgLd8FdIhv4SxugKZpYAJjEac9AVCygSmVfQxfjWoqn9Ge2RjiA1leKYbDPi0XcATP3rncgIGnjPwKB1sV99%2FLP1ZxN7K%2BvnvZ89jh5e7x7OK4yNCMOAMAB%2BEr7Fuuuws4SuBhgj5kRAmD1kuJo6Tb%2BWNzTrqr8K8sAzWiZEACWuGmlIS4EwFfdt1h3GFcX1nCbtHoQtZcUFw%2FePv%2BKgsT%2F9av7pxWCWIW9Y0qnxCYrZQv7xfqtXNePl%2BQ72izHZx3tLcZ3YC3A9LmaK8fHayR%2BXI5PZlUdSOX4RmY2VMpgZdBSNT5d45bKG86XPFOrGp%2FBnVUuL24%2FuecoHXgRvhqsLaeL0k2%2BAWrVgnv62ILC3jvkQ1u6lBv01RNiZqb1B31lq%2BwNoOvOEHSHov5NUZqNFFeVXSmHBIpqru9ThFR%2BLtO4gvhsiBE0HAnqZarfAJZ9xfrwrZlBqdZVtAPaK76NaWRpRDEsXTsaOFFg3Cq%2BQb9KO8njdqNiNC7BiVO2uIcOFMiEThvMAFM%2B7QGACiB0TRWCwLjLfIOOlTahx%2B1HxYofE2fB0K5UA6x0QghY0uePSAiBBCFMRRDCrBBOA4MQNQRg1hUAMUK3ehXDcKugRcTmJ06EkRK6FDNhADB1U1IbLgywVcEAZbPa9hkaYQBaRynGzj6%2FON7sn6%2B%2B2zcIkJTVRhAARdsl0tpaWNaufdK03NJ2yz4qtbhdP6q0uF3Y6D6%2FGf3HW9inL6i59fHrO364QF7qvi2QFsjrhpVbUzzKrQ1qaoW8mWvYyDdcb4U8bzg2DWe5cdcMbA96f3sextC2HobnGRzynSu0Yh%2FeysrVpS8Ht18Vw7Rxz%2BM97Y4tIEw9kLyTsBy00G0j5zqGZrlx05oBwA6cKoRgez9ilxP8MusGvypmJoAfu4th0yQz4eT8m75TinDyE5rbESQ1TTkgRnnr7euaF4xSMj8BsOwpzR2YHVDGYznEkJaFKw0ximwII2IMxKhrhhhdhwm6RYyBxBRIQowOdC3GW7%2F6j0QYDaueCAOYGVAGJNZJjLEqhIExGbJBFysxGZLAogM5izHcP6LlNQv3ljoCQhfIpE40AUz7lA6JlSbWkkWPExgzIxv0sRIzIwknOpCzGOL7P9%2BLfjxcxo7nOUSZHQlY7UQUsOTPF4FsT9OFsUtl%2Fu6cH4Srdw4XvayVOqNh5mrW1WhLrtQx8vVyoSGBRWG08mG0Gt43NVZanrN%2BH8WxM%2FBRs%2BaqF1gNFw6l5JuGFa1w1Ayw6ilPB5gZUNSs%2FDwXAKBQZTGOhTFU1qBfpbqhuN2oGCqLaeLKmaOc1QKsc%2BIHYMKnOBkqflBlqY2FMTbWoF%2BlqqG43agYG4v54W65J6%2B3n%2FxANUPVEj4fzxI%2F4OAHZRbX2BhrhjbnWO26%2BiCA6FbOYoDuJFgkRYiSNTYEEVK1ThABTPxUcRQXRHCdooeInlUctaniKG5HKUbd%2FoiWP%2FwvTu8qjtaWMkFAV9qeTGf%2FHb8tz%2F7ULx4upn%2Bc%2FzZeHo5pYSQuBODdPR4EKJSdOOxSBAB2Gxl1%2F0hdpDhKQp9%2F052UqfsHhrbiChxB0JsluGN1eNHyZlV9OvA3y3C3UpZ8PD7arEo%2BNEtWJV8flKtKXlSIPH0tIMqMy6xuCarMuJWrFWwNWiozni9SPMg3XK%2FMODyzL7H%2BqA2D3rJmq5wtV9xhoHG7tUrbrcyScaDsdmTnMgfzPWdTdjvScg0bitutuH4rqSpz6i3ud3IowlxYmJXsTXu7k7DLbgQKPj%2FWFtdR7ZdTn6eBJEVT6zKcYqmyQ02QYFZmdqcS4eTLNlfPIrVMOchA1Vs6kLU4wU6F7FuQPZVxAWYHFWb8e00Y8grZyyIMVXJnh%2BJEPopiHM152dQoCS7QOlVxHdV5GMxP44ckvGhU%2BIQXwCyhQl2EXuMFvAkMVVJrhyUiqeDWtDToYi1iC9QeVYwQ3gUYyAKZzIkkgOmeNvXFShLKJNkOK4Tj%2BoAStJkvapfKTXBDzsmCBaxlOgALnWACmPIpvaaV9BppBKFKhu2oQrgNDEHUEMCINuPF7SjF2FyCCP9eHSJkhC6lTBAATNtUKxwXBODLsd0hPIzFwut4Tto4F7ejFCNol57%2FI3RZ%2BHmOshhXl2ImDIClbh7nlpFy95uFwaE5d8faJ00bHW0m3unj4VEjabQbicMxIUyDx8B3ZptJw8ATbccSCzeBStgbDM1PxnbqnDnI2WBTOXvGeLvhtWXIydnjDccW4Sw37poh7UGvcPfDmIPcWxwPDv7O5VsxdKvGm4k%2Fl74c3P60xNxqG%2F5Uj92pvuVOjY%2BqEhQ4vUOzmz92sZDcqa2oO7Vyacgmn0duvm6BQr7UGuV9qd2AL823YgzIl47FcMEqSy1i853DLkorlzNA083Rdlq5zlNhP5p%2FWo%2BkoE5AjanAcNtrqVJLrj8Jqkpa%2BVicfM%2FSyvWdUuxD6pfMkS3NlXaga3Fu%2F5ZFlFbesOppUhWYGVRIuuk1YchLK5dFGKqklY%2FFRJmMMLqOtnZMGBIne4gwOtB1cam5ZE6AMKNh6RNmALMFqo%2FXyhIuWWyhSE65ycMZfVnBNaZqd5j9pKntqHYXM8Mf0RI6NEATM1EAMHWL4YyT50UsNhbup4EjVFGywyUvKSJm5Va7WPqgXHe%2BPgm2P6dUgPLTVYfrMLNUioKt34cYLUBR%2BrCxzYwzS5RDmrQnVweKFif%2Bk75YjZkpwLKn%2FbuA2UGJjXyIKI7qzkxJIwpFol6mVqFMFpiZqQb96pBwArUbFSf4r5w5ytIDgEVO8ABL9XxbKoIHHPCgSlhLr1AWqwfwkJohwQNaN2oIcr5gvrsnKtBLfKgtc8IHYLqn9B5U%2BKBKfWRTrxBG6wM%2FSMzlIX7oQM5idO4k8Y%2F30VVACCFV6YQQwKRP%2BTu4EEKRAsmmXiFuBgYh6nhOidk4xAgd6FWMt53NHW%2F22XVRTjN0KWWCgK60PfjXXzd348FnQ2OvfzsXlvH5wjnWPvbEm1WuYnV40fJmVeAp8DcrXbVR7Uv7pOnb1b6G%2FLD5Yl%2Bus5iuvtE7RjdX1yub3Pmwqpc%2BkBj7hlTVa8i5hi%2FdaKmo1yjXrpVvt15NL2jmXyIW9HGRu5ZMX7O3TD%2BWxOG2r6lg%2B7rEXT4h2f6Ih8a57efr6jVk%2B%2BNcuxbien4jY%2FtZhpZ58Fcu3YidgXI%2FqvkVOlQxGgnUoW5z1AeuVCW3aajqNnVtaGzbPJ9naboQqp5r2M43jMhx6hr3%2Bvwt2oeXKC3fipVxZ49dZ4mAtmzXmTrIsp7z0EL8G7X%2Fc8PXxHD2Od4tB9pFuX5ewfFDr2pLLB4AyakOebM8KZEzYtPjUHO7Xcy1pfOYaGZ7zTTKovq4T5Wlv%2F%2Fn2H2avty%2BDEbRq3v%2F7dqbaMcldqACNLd3CI9CnsIrcp2FP5PE3TNBeU4%2Bc5YZ5KCtGbxcu8J2AGhn8ArVUyKGL4%2BaAp%2FV2anD3J67N%2FcbeQHubPPQgTzW%2BEB0WHokqqblW3wzo2Wua27Y8u1cu8N8u4iYycpxJ9%2FkRCoz5RsZZcdSGxkOO2hkbPWe%2Fkqssel2OL0R2TG2OoZRrPW2orpdUGDxDVNnpX5vsB5Rt9wb2Pl21eLAEiUBYAQeDhrogQ48aKVtfKyojdsdjfXyY0zJYz3ebjvEl5vxs6wmiC%2FXyMDoPSeVqC4Nw3EmI%2BjhlvM0Rh9sHaz68jdb4sJqSP5U1zpyqLqukEfVNXv7YYZWE%2FsD51vRawSM8flUpk2vby8uB9Hfy%2BEvJ%2FSvfhuDqquKocDo2Nwfjt3rT1fh3W2faurjD7zq6gjwHuyqhnV1M09b%2BerFTTnYQZ7A8g1jcrD5tzgwhg042Hwr9rBPDnYynf13%2FLY8%2B1O%2FeLiY%2FnH%2B23h5aD%2B0Wyfyo9yAv2yAR6%2Bd%2BQnUfdr5FdptjffzK7SVie1OzL9HX05%2F2ePHm6sfzy%2FX3uW3ZdXsjJasvO46uCr%2BAOpEnqYoF9n5rjafAdmUYedxLN8udMNeXRsGQbTnSv4tQudpehm4LLnr%2FwA%3D%3C%2Fdiagram%3E%3Cdiagram%20name%3D%22v2Official%20ERD%22%20id%3D%22jasQtF-PAra0dogdg8P5%22%3E7Z1bc6M42sc%2FTS6TQpx9mUOfpk%2FpJPvu7NxMkUDHzNqQxSSd9Kd%2FhY2wkTCWMAghqWqqJnY7MpH%2B0k969BxOrMvl64cseJp%2FTcNocWIa4euJdXVimmDmO%2FB%2FxTtvm3dM0zU27zxmcVh%2BavvGbfw7Kt9EH3uOw2hV%2B2Cepos8fqq%2F%2BZAmSfSQ194Lsiz9Vf%2FYz3RR%2F9an4DEi3rh9CBbku%2F%2BOw3y%2Bedd3jO37H6P4cY6%2BGRjlvywD9OHyjdU8CNNfO29Z706syyxN881Py9fLaFH0HuqXf5yLV%2Ffuj9kMJPf%2FtY1vX4z3xummsfcsv1L9CVmU5P02bW6afgkWz2V%2FvVs%2BLdK3KCr%2F5vwNdST885%2BKH%2FPgvnjrYpUHWV6Ot2XAN%2BAI5kGcRBl8A6xfLxbB0ypef3zzzjxehF%2BCt%2FQ5Rw2hVxc%2F49covNkMd%2FFZOPJfYGPFy6Lxn7Dx2%2FJhin8OFvFjAn9%2BgF1SfONFFq3gs3wJVnn5iXm%2BXJQ%2FUvZf2c8vUZZHrzvqKfvzQ5Quozx7gx95xVRUTg4Pif7XVmnAL9%2Bb76jMQm8Gpbofq7a3Iwh%2FKAeRYUAtYkDbB%2FKmUPTFPM3i38XwLcru3h3c9etf8XIRJHCqBCH21kW6XhvWgxQvFpfpIi0UkKRJRIig%2BFCYpU93QfYY5eUbT2mc5Ot%2BcC7gf7BnLo0z58SBz3oJX4Pta%2Fhf8fEsv0yTVZ5BsRVtRHDMf0XFuF%2Fk6VPZ6CL6idrPyn4vfr5P8zxdssnCpJZFKQOLUgXOUCKwCRFcf94rA%2Fj353GwuIGLb5A8LjaDtl6Lg%2B2gNYxsY19X%2FYt3PD5%2FU9idPxfr5XQeh2EE5%2FLFr3mcR7dPwUPxoV%2BQR51msUU9XDvjYzEOT9nYttOYWwsWcN1KghzOn%2BckXBFjXj1ndxk4exf3T1cjywEt35vPXqzgoMfJ45fNb7qYXhxB9PK6f3o7veqHqjkOAnI1TKhgYgwIE592SzEUTDx6EfBZO6IwRu0NSxKXeqwOrP1tY8NOErw1DguBT2jgW7Dcf0QQhCEiKIORGd2VQtUcB6nMNDNGZwagPocOBQ30AApSY0Y9WrJTAwBCBB%2BiJIwydcFBLw7lwAFIW6QmB3dy2KOTg8GCKQw5KIcD0I%2BH9GwgbZQXcZbPr%2BC3ThEP%2FStAPQCQ9koNAO4A8EYHAIPVURgA9LM7rGaA5gMgzY6XxSR%2FyL%2BlU%2BQDd4Gohw%2FSSKnxwRsfpjE6PhgMkMLgg3b6%2B9TjITsfkFNW7XI7iBfnYThJ81L%2FClAOACZpbtQA4A4AWuenwQCARn1HBu9F8n4yBt0hmvROjNIjgrQkXmRB8jCflveTMHpRDyikjVIDhTtQ3NGBwmCXnNqJopK4xoVJ2h3%2FSO%2Fv3p5kvo9gGH%2F1ln8Gz0e9%2FA%2B1%2FM9GX%2F4Z7IrSnSY86vGSHg%2BkXRHiQR8lOolFOZagr9QsGZElljk2SywGE%2BXUjhKVxDUrLNIEWfg1fYyzKBSdF3wUoB4AdGC2AABwRgcAg0lxcgDoLfJ6%2BgAgTYZfgyR4jDLxDwx8FKAeACjcGqPwMUJnNbiYztPHNAkW77bvbk5zm9UfmPVTWpSE50W%2BHPjy3c3vKEvvUqi5N8QM%2FN%2B%2Br7WTFX9vVJFjBy7%2BpsndV69x%2FmfxZWcGAOXr%2FxS%2Feea5Tvn66hUtK8WLt50X11EWwz5cZ4dZv5fA%2Fvyzenb4YtOWg15um1q%2FQm2xsWGVPmcPUduwbD6XI%2B4dknYxRLRLW5VKaFd61ZtZtAjy%2BGW3tVZ9Xxc03s4c0y0NElViJkzTmz%2B9%2FK2tnomGLKwhgDe06RuiISio4G3nY%2BVuYf8Do3bL73H91sfCP265DjYtNw%2FQ7yQlTb6bK%2BP2vRo2dRTNhlSlKihHzKbdbQ2XCMfSPqF0u26WfCn0Zj7GfEj2YDIgbbcqJUTqzWu0bYDY9%2BV4axw2YegrJ%2B4SNHBCpMF8TLvrh6o5HgLSPqa9m3HYgTJ6jJpNGnhHXj%2B4xajZ2sG06op9DqZqJ0diUIhyViBbu5QKwA8wepSaPUWf0p6WB%2B1yWnUFaRO%2BzbMoyosotmi1Upgh2i91f99ov1QRGDJ6oJstnGMqP4Zov9SqKxpsmymc2nAkQ5UPIdoZdW%2FfoMfQABkVIKMHtjkSe6NWGteEcEhj5WWcv02RDf0PvnqLv3ZEFWHxHz2szZHYE9XRnqhVV5CGxttc7hyrDKOv3uqvS%2FoIsPqboweiOVOs6kM7%2F3XdnqorSBPhpLOo9q8A9QjQYC8M3pZR6dytfZxbdYJXfKW14VQuS72v5OgJNNH7c3Gu5kjvLs6DAd0lbXkKuThXk0Aj3yWtfXB9v4l%2Bik98fi7ODHpRboPgagNh70dEdqCMfjnkkvbB95%2BBQEQxhl0htAmx6grShAiJUiRGnFbUjDCCUQ8p2uooAFJGv3JySaPj%2B88TsDv2tEBos2TVFaRZ8t3yaZG%2BRRNAihAKUQ8hDGV9NEKGQggY%2FeIK2VFrDLHUOZbQD5nsEPFIi%2BdtsIi%2BZ6H4DBFSMMoxxSNNpZop%2FJkyelJGT%2BKS45XGNTA80qJ5HcSh5BXHGQSgHgB0xXERAOCPDoAplhynnf%2B6pnjVFaT9sQDA%2BXK%2F65ME678uGb6%2Fb3R6SAHWfxOMvv4zGBcnt%2F7rmuGoK5CmSNfXIiTmeZLpVvpXgXIU8HVORxEoYI9NAZ%2FBHDg1ClQa1xTwSWtfSYHLOXxqmW1BDCqQiALp%2FT%2Fwq2Fri%2BA%2BWtRGfFvdt%2FRoW2fu3Jn67Uqyd%2F68Uno6fKZNTgCgvT6jT9KMTV9VaQu39m2nwJj55BeaQysQvixFyLJOaeMk3baEQbA%2BfVpBUQJ7fNI2qVBgj6%2Btl1VXNFkvJ%2BiGPWxkD4NgJNriUCpImzt7P%2BiyE2X0yB5%2FitbOnlYHbQxFXYF21CRO1K5ewCAR5QAyo7CU0laohLLYV6ByGSRhU3nKMFjN1y9Q9ck%2F0fpRVaI8UIdy2JqTKDryUM1JtEc5WHQSxZQJU3TSQqKrDtSmVW%2BDtuqkbWMtmShj74Gyk71pmbT2lu7DUda%2BK9KmnOJf8UwoFippcXBfgz7Y%2B8ZmpiPXe7eYVNOk%2F2qPg8mgwc9THYvJrLfA9bYBYt%2Fi4q3x2K405L6cYoTIsBYTBsGwlnvsLCCq5ngoSAeu924xYSeKR7uzGMpiMptitsyeVgcdtV51BWk%2BLXEy1RAS7hJRz2Kiw9YFAAgwxiZIZYBRESE6Zn0rg71B64ULssIM0WHqLaIhTZUaIvwhYo0PEdLAOX5ORm4c2c4DDRJgkEbOTe158U1bYihEQYxoh1IRMOKOjxHhwt355UPazgINEWCQds31%2FXcRCJNEC9FBIqZqFASL9isVASyz8cFCWjvHT%2FDLc5XQ7qVVX6BAr93SY8%2BrXPzjiZh6UY8qQIflC0AV0xydKoDB%2FCkdUgC9Y6b8SCHNn8XN%2B128jC7T5dMiyqNQ06WLdBSkC0P5c02XwejijE8X0iqqUkL57UTQgAGANIxeBqt5HGX62NJNMgqChcFXVINlMLD444OFwSoqH1U86hGTnyqkVbTw%2BVrdBa8aKl0UIzNUZpf%2B5bX51zfX%2BW%2F06P0Inj7%2FyE91FIoASLEAR6Q0ykC4KJRBlofWGaAUThp7gtxVfMiCJLxLi3muIk6O1IvMMGkmLUp3uGtGjRYx7Ku3tffHp%2BRn2s4XjCWKZkBwrHJjVwLCb9CQ2ZSNwzIGO3SY%2BtKMbqvAIoxqwhyTA2GPEgYTAnlpViRBuBzfX5xfKoTtbGDaNvS77Auz6pOXZzoZwlGSea0PsAo7B31p1vtBtANdGjJI8qYLQyiBbFFIZqcLM0mhQlqm4FHispjrD%2Fm3%2FacIQbgihkoU5Ii%2BIxOBIw3FF3hzRLg7Mo4c6XRFJilHyCsyyJHbPIui%2FDwMs2g1yeJs%2FJWiHkvQd2qWjMqS6lfGg4nFYPsUBia0I2LRj4j0tLBI2yakxXW6grMknGQitgFUoCAJdPpnIUjQIDneJGCwU06PBJ0SPEtKAtIOWdif4vxNagowZ2xWiQLaWU4ICvjjU0A4f7k%2BKdDJI05SCpBWxLX1aKJZmQeQgIIY0ImXRcCACUbHACo0JVRSmj5J0Cm9spwksEkTYOnaWiQICJI38X2dOElBPSLYpNVQE2EEItjjE4HBUDg5HFQy1zgANmkHvJvHWXgNB3UT6zBNH6UBtKAgD3Q6ZCF44I3PA%2BHyIffJg075jiXlAWkRhMO0gA8dhRdSXxkwqEA9Ejh7T41v5bGxHQwYBBQNh6zCEqqiWw22n8aA%2BUoI%2FS%2Ftjj700UGeRRkOvTYYa0IPx3iHPPMpVBR6Ow%2BUSqCwpy%2FIUyG23ItvJeQYEckgHfVyKTj6ENn%2FIZKdLw0Rkbz5MsUzZF8rhC6ps%2B0L8oiJweVbsJykWwJ%2FsSiIE11JRwScNARG8sYJg%2FuKdDjRVXSqvkAbm5rFchNd%2F7xUmSK6ck6LaHQSMBEo0hQSyRkjLoP5UzaMuPQDJj9GSOvnu2UQL3RsPYNKFAQJRRzd2hK5YQQw62MQJeF5lq2H7d0NFMZd%2BrW4SyvBsv2nZZCE39e6yYq%2FJUJwiV7j%2FE%2FUFvz5P8X7Z0756up152NXiD5hsJqvG9g8AOyAP9HHihc7TRQvt22sX6FGaMVTdkb6nD1Erewouy1HxNv7SeTBGYWPEe2SBZq8Jao3s2gR5PHLbmutmrwuQLxVOzCQvx7y1zDOUPwZamXTAeUvbmVItDUz600BD9P0pn%2BIhvrTM2mtLWpSwoaz91EU3gcP%2F23fKJ3om96TomgTqI8j7UWvNeBOR4f99X%2FR69Lb6oS56HVJA6xKF72ukpny9%2FQFaU1Fy7y%2B4e2mGQX3wDqQsH9jCjtYxr%2FhRZtVJW0pnWIM5aSKR1pXEVWuJhpxzl8l6nHEYzDFao4MxpHxr3Y90hY7flEOfiip5oFGCfBI%2B%2BpFFiQPc%2FGPJ2IoREGMaIdTETAiwN2up7DHqac9Trd9QRo836dpeBPkcMVWGSPa0bSlc7SjqRAYaYqy5YyRKXqaUq8B2pW06gskq10foOQlztJkCXtxurgYQAvq8cDXLqNC8KAhay5nHvgMdsrJ8cCn93WRnwcNBWGj7CV%2BiORnAYMOFGSBrvIqAguaUufyZsEUy7xSrwGd6rhKygLSlnidqUEC5jqtKpFA12kVggQNKXN5k2CKhVqp14BOlVglJQFpDryJXuLol9QQYC6wqhAEkMeMhsC4EGjIk8sZArMGz0eZKmlUStccADPSDHgbLKLvWThN36QBNKAgCnSFVRFQYBnjo4C0Eb7%2FbEmEgk5FViVFQXPUtOQcYK6xqhIHKFwPafNC%2FI6ylDkxRMe8DvXkECi7xJlhWCe7GSbM2eykLcdEkZwBhWZCXMzTxzQJFu%2B2764%2Ffh1lMezpdV4AFh6UPXkoqQQqenk4pwTypxYmp4SNpZQ4ra46WDNKuC7ekou1NHRKiRlpIi33yVH2KY%2F252ibYD6JJsdMNlXTr6YAALM2tBZKHLpbJ8YCDQui0yLII3c92muSfv%2FbwWV3Rm%2BH259volEUbYvUkZogDaZFvonL8WO6aLJO9DS7Z538LN0GxlDQZM%2FuiKY1DpujqnbVFC0nNEkn%2BEvmtT7Au9sUj3GuH9hg0zTHRUPaHZMONB0zXrGDBm0qDu4%2BhjK5mAZpkZ1EYqN%2B1ovtlDgaMd0P4DStcVkeGlw3vxfbfPHLmfIjDItiGAlzhAmHpjkuEtJOnoIdZfzxCUMaeidxv9vXetHJIVRSwpAW3q9R8iz%2BCUYMfSjIE%2B0qKhhPABgfKFP0Gu1rtejkVCopTUgb6Y98f1FlBVDC7HCqEEoAaUDVKDnWy6gDP%2BzR%2BQEYzKDC8IN2DahkrgFhorvenYGubtBv8yB%2FnmSJmgGUoCANtM%2BpEDTwxqcBg8lyejTo5HAqKQ1IO2RZQ%2FkuvSv6W%2FxrDk5SUBAHus6PCDioLtdGxEFD3kuJQhC2StdEMAFpSDxfreAaHIV3crOAuTSPSizQpXmEYIE1OgtMBpPh9EDQqfiOnCAwSYvgVbx6gF%2BYS40B5so6UmJgdulfXpt%2FfXOd%2F0aP3o%2Fg6fOP%2FFTbh4SAgMsRAo06kMM61CpxpQDQ2BOkZegW%2FhZcaNfp6u4vop9pFlVEuDRPzg05uHCkLmSmwp6dAnmlpLkwAhdm4x8OGHYIApPhgMyVYsOeviC3ADU6nP%2BEzyAjHI6Whsx4aCSndjESAQ6WOfahQQ4Ho1aJKwWGxp4gd4K3z%2Fd5WkxjKZb%2FI4df5sV%2FDxkp8trT5i%2BC486cvghlHjrZzTrknLTmHOqY8uif5%2BUTircMsgf0TvmH%2BQzLe9lDh%2FISocRZh%2FMSoQwAwuQl8q1SFyWe7K5piYCBVI1aMrCWBk5LBGczofAifqt9k4NtaATPRsSmW4bMQ9VgtWQeaky3WGXKG%2BAMq8Or6DasDMrYTpJj0g01KsEeTgikJ8QkskD0U9R1Ow%2BOzsvZNkTt2xia1rjsYkj3h2nE6NLkgOCvF8Y0nt31Q9UcDwFZ2grSuxWkA1Q8yu3FcBZya4pmkJ5WCKuTnUTKTM%2BmRVpKCqJcRasH0ZkihkKUSwVtWtr9RgSGAGN8iEzRAaevJUKXC9j2BWlcvQhW0foKVmWK6IICLZ2jo7qEoIg1PkUYDJ3CUIR6DegU0iUpJkhD5tp6lUfLy0BqZ34GFShIAh3TJQQJ3NFJgIrEyEmCTjFdcpLAJq2P5y9BvPj7UxJKzQHmoC6FOGCTdiX4yPHT%2FiPkBF0YuBZUcu1y%2FUR32Z5Ft8QDc7g1Xmchp6d9B6ODTW90YPRwGJD7pP1IuYJK24mhlPfunr6Yai5yjrUuGPTS4u5r9ruXoGmOi4C08xy98xwXyjSJjzNlmv3oLscveMGTMjpH%2BbYvSMPTp%2BRRQ6abXNSDDDo4a8iIcpSZjQ4ZZ4p%2BdT0tFtV80GwxHZPQwY%2F87TrKWmNyBAGMGBpRECjksRduSDI4g6OWZGcTtIJSqoXd4ul75Z6uisKjtXiinc4ASNBnUfqzKPU6Mlgg14B7A7UDuZzeArmmfwHqSHz05C8X9e5KXX307N9nhp0p4xs13SmeN3taIdxO5005geKS500IlG%2FBUmUHfAaBKIgQHcYlAkJoTZbmYOdTV%2BEoLre3KK628WFHCN4alwWB9MKBCLl7a3HAUwAhg8VwdRcMVXM8FOORmw6NEP4IAagu1njHEI9hMyEMQ2hHpJK5PmeYHrlZ%2BFcS599%2FfoUT4zmbJCoG0IF6xwmPIsFmkfYRWQ9hX8f52806fWOavNv%2BywWHNJzG2Wx2spuJ07PckyFycRaNXEdZDLt3fQ3HAALa7JsV%2BQ%2Bm36xcRoVJvwkclMC9RJmDJ8Wizr%2FpYi3ZeEtD59%2F0KALaeUj7pEOGWcCm6r4V7NAr2BNNwb6L3Tjjiy%2B1gn0Da8nkrWDyonqdCypuyQU1QecDriFYtm%2FWBtWlzbdg4pmIe9ysk%2FfQ7eOr8rGtg83AY6%2BKMPo9kkdeSJfO8XulIY6DQk%2B%2BadW80P6LJlKfjsDqRS%2Fq%2BTL6DBfTSkOmYwQWO2TGd1ZAe6GpOcD1tF749NtH%2BflCGozfZ%2BnyCn6rJkwXxShIGJ1JQrBjzPjhVz6FFVq4e6i%2BFgudQGLbF6Qx9i6dAlzE0IeCMNFBOoLBpJLGiDRhMJRKRxOdKGLbF6RxtLgUmaprHH%2BFqMcTtBXWPOnRNa4DRJzRITKbYoQO7YhUMteUMGekdbOgxP%2BtX04QEwNoQEEO6CgbITjgj8%2BBKYbZUK8BneJoJOUAaYP8GiffszDKzpdSl7lgUIGCJOjRP%2FR3lKVdHURPjTPD8E92vUTBzDsZxPt5x98bQmKePqZJsNj19RbLP3pWnvPF8S61PesMFX1GvmI2Jl5qB9Pq7ytbqnZuBxxMobaCt52PlfuD%2FU9teUb9kUv27X0y%2FBfguQ2bg5tn6HlGUth8ecxIACckqE1I81A4QsO06erMfXiSjjYhLcMUbUI6mJO25XSdja7JaTa6Jj4b3UNPhs1Gm8ts3F8dbO%2BeSXuft%2B6ugI2NJPAoj0EDFgCZkcb09vFV%2BUTc4aZtRu%2BVKor3uYVWZ6ULgFQTQ5%2BhLYM0mt9GufY%2B7yYX5Q7b1cZRQ2YY73N2yIzufW4ZpCleufof24mhIWMZpEUenTg0abppRkHSMHgcK00algs%2BdryM7npuGRT2%2Fane721VrrFhGc1pEgps%2FMjfROcGJxUoCALSfnn5vIKjFGXtQJiW%2FZJSK8y2Sseum8MdYNOt6AAl3hlgSdemSvpTJPUiMkH7JMrFMrXA5X6SoG7ngU6AaAHSMlks83IcGPnrRblEiRbQpsneD4wdoEJrjzQH212AKebM7WmFqCbB8XnXW8aHnSh4a1wWBNIMqXbdDhZ1sCZd76wWqua4yEUbHEXgB8owMOKhZIoGx75WiE72SElPJKQ98kOUhC1mJwUIwmypVOkEovO%2FikAQAMZHCIOBUzqEdMr2KidC0B3KrlGrmOkP%2BbdUZYowJ3hViCImaQjVFBmBIvboFEEDPymK0I6ISX%2BTKT8mSIvlu2UQL87DcJKHjQE0oCAHKOxRPKIoi7BmFDVZRlG6B8rfNERRhsFqvm66iqLsEPdMqSvagMly1h0MlwRAtPpOPro4R5cueIwjbbQk0RCgq43DGiyJf4%2FnWJ2ea9BQScuksPwdDuvlEtpcm5Dt07E29YaZQsJMDGCgYoJIaACr30odRwwwCTo%2BNsf6iiM2fGxuuE77k%2BG%2FgJ5s4MlBEdXf9%2BTYTAHauXFELUGUwAPUJxbwTPqpxSHnxowWWah4mjAT0zPqmrXwMlzUxLLqDQ0V3u%2Fj0f1%2Be3Q%2F%2Fnkw40IsCpNzv9VBj9pKMlCra9qNIWlHPf0AEqU488%2Bpy9Puml%2FDxxoiEnUMXE3RMils6%2F1x6IiKoUX6GaumeM9ql%2FxQ56Z%2FnpdPqDtKK8D6rc3fX2im932hTzlTkJKFmSiOXde3j2fsoJ0oLtaQhzfUE6gcjKwok86%2B58I%2F7xtG6%2Bc977jPzxweIES7AnF3pzvmFLO2KhT1aXklpdpdCYLsob4QgAEWAnpkCr8SVFvYY1cCF29oaGRaFFdKYtg1Ou0QGWk4qD0EuehTbBFFs524fW0R8b0m7RaRmXzYUdBxDpAP%2BzwqBD4wmcypTL5iw%2BrVJmBRi4IXmmrTsp1Ts%2FHuAlxXsFkLjL6mLQCc5i0wXIyszoE8pvgvAJfLzKUIJBFj5taxObParZats3ZtBa3PXGs94VpNqMUrcfINowLS4kxSC0cVfplAPUltHGJDXUtYPk5Lr%2F3J8F9wvQP3GLbjt%2F3CULOaol5AvybTY2xIvPfD%2FMFLaykCwiUtdvFL6s4bZvyqkbdN1aLwNRlhThx7o9f5lrx9FvijnhsN0eDmYkRycW8p6mmAQxJvqL9pEL4PXmPn4t3fy5vgn5dvT1%2F%2B%2FvDHaUPg8EUWJA%2FzT8kLHOY025%2BEaIIJZ3gmzPZn9ZE9pU%2BYjS%2BFXdxrm0dblzmn97NudfJvn0xCpqNpfmSSgjJny26fFUolpGvuCtIb8FPyKEdCGu5ikTlvXXPf6MLndHg5mAevJ7zwTJTd%2FMikr5bMebLbZ4XGS0OMcHna0ITpohflCNMQXKwJM%2BYBhmeu7GZFTLEkej8LRTUZNFjQoO%2Bo4Ef%2B9j35GCSh6GQRQiAyk%2BSDl378vLxfXHgvf65Wszvrb3t2yrBsKA%2BSwehRiYEHPhplQK4b4tODHI5WhStFh8aeIO84vsaJNPUVjhx8mVf%2BZipyjXg%2BwhGi7uEE3IEKqw8fwtnk79B69BXnnhdzwjvtfM%2BLNwQoL3pZfZhMLCbT9ttdmPDPn%2FYcw9k8BbnGVXeZgl0LoFcRL7PZrDZ9TYM9gUjP09B0GrwJWy85hJmHwOwakgYsXOCAMhSFeebhOQpsq903n%2FiFU3Ccd356%2Fw9EL2xvEdxHi02bt89PT4sYJXCtrK07cw7N1MT7%2B%2Bl%2Fn5Pf5%2BAx%2FZ2Zr%2F4fwcdT0975g8upLYcvCNvkot9ImQ6%2BAqNioUP4fcCX5ag3L7%2FNg6rTftPf05E6aZkoovp%2BND8yuRGWsxJR%2ByRQKhlfc1eQ2zGEDTku5rgrRubUfY194zAUEFAaKc37%2Bp6QwtPfo1kGDB5A%2FIFiDLo8VFNAA8Uh3X4QUKZQjkhIzSiHFJoMDhopjcvcUHzh6e3RrAlTNL4MUWSifTpovDQEqe%2BUmHAXxfS5z%2BBPj3nVv%2BLSRgjJyEyXO9eIbz58z8C%2Frj5%2BW50%2BGz%2FDv7T%2FBwNcBuMJV%2F%2BPRhkIx5NB9p6tM0ApnjT2BImT4rSyLkchOjxElIvMLGlc%2BLyGKJfn7GEerKLvWVvxxAlemzWJCRGu%2BJK9ourpdg1gYdWWB84cWoh4Z85%2BrR11MPEYqt8pv5%2FYkRMmHYbdhsdc3mr0Czf0ZYoEW7dPFKXcUpvlQJ5B5Lp2464YmX1ZmyWky%2B8Kj53RL%2BV8cm8icxB2%2B0TR2JmRuxB1c3wcKxaZidMcS0OqRxPnSMMpO1N4XsQ1y4Dcu5ZMsUZeRygu5OrYIcaoXfdKAaS5K0ij%2BVSyeBxSg9O%2FGmQmRDM9Ge5U%2BieEqDjAzKi11KvFi%2BsgL0ZmsyIZe84ohdzih2BxXup8CTW8CYUiZw769ffkPw2GJer7PFTQrP%2BzDnmXMwmf9p42rib1iB3AStsAsUMKb43HKkSmOkV3Mlfwi0UHFcfDDr1mGFHWXUNUzfHY6WivdgEOO8CejX3aEc6tvb80IVYnt3U5DzakNf0qWsSwZ6JwCtDgowLlDjQAlarQJxpcnvKfaDyO7NmjPuEyHPI70GynnlJ02tMXTYaVjbc7ZEFSU4T7v%2Bc0L4fq9Ff54OfwI0maLYsF6WL7GeQo%2Fyl5hKCL4Yisc2NtWoaPuml8Gs70YkpTQWT2cHICGplTRKZp0CJzKCMgYClEIB8y%2BzvQ9WoExFvjsgw1%2BUexItM2jEZeXgSr6DqL4RhpVvanSVZW9mtkHFak7uL83%2B6N4d%2FOv31Nvzz%2B9T6yv59qz3EBbIym7fM75zXKgMXIILCNsVXiSh3imke5wW%2Fm9vk%2BT4uJLDgvuAhAwaMSRcl19ayLSvhLVFXbR7QuMpQEle%2BoRF%2BxQCIw7emLpqSgPVoXb%2FMgf17po1KPmpSZlc0ZrO0mEzgmHR6Z%2Bc8MAyu07rgn1KXWm3J9H053TgrouGzgpmjJ9oHplDBCAbs%2BFq5Nn%2FTbtmot2Xhq5Z5yfgMLFfVGX1Qe4%2FY%2FGf4LTu0Xjs63b5uPn%2BeLILgGFy8fvr%2F8%2FfGv%2FNMpxdFyV39QYHH%2BdrMezTQZterFgaT5HdP0d59KrYlhd%2FPqt4ayCjPlTMusTxSna4ELKDxsylEWuOhL4xS35MJoHHTRuMFL4%2FTFI1qj58QRuWnX19zuIq%2BqtqCWKLnSm4caxWlNjNpFZwbcyO6q3CvGh7EAiiDab6pf1BoWJIz0sWQ5wPDcbso%2FxYRfFUbqvYgKNsGMQzVUsL8QFYrZP4WdY3%2FD9mollVi3bPBllhZH4%2B3H4Qlx%2FjUNi5ot7%2F4f%3C%2Fdiagram%3E%3Cdiagram%20id%3D%224ZGgKWcJWm9zKwuppqwg%22%20name%3D%22Page-6%22%3E7Z1dc5s4FIZ%2FjS%2FTMfgruSxOm%2B82jbO7zdWOYmTDFiMq5Njur68EEjYGE%2BiCQVgzmak5kZEtPeecN0hH7fTGi%2FUVBp71gEzodPSuue70Lju6rl10u%2FQfZtmElsFwGBrm2DZ5o61hYv%2BC3MjfN1%2FaJvRjDQlCDrG9uHGKXBdOScwGMEareLMZcuK9emAOE4bJFDhJ6z%2B2SazQej7obu3X0J5bomdNfOEFEI25wbeAiVY7pt6njk4Ho7cGnZ7RYbb4T2%2BMESLvNhONF%2BsxdNjgi3EN%2B%2F385zeIvjuGLinrnt%2BvNKtrTH54%2FuXlrOc%2BXOv%2FnfGO3oCz5MNuYOBOLT5wZCNmg46hx14S8MpMhk8AJhyaHu3PoBgQYLsQU4MWXDsO8Hw7aB5aLNsx78EGLYm4kbgyZvYamk8hM6wtxeee3oxdspvP6M0n%2FMOwXwPHnrv09ZQOD%2BvRwNCnn%2BUe%2BIS3sMjC4S%2BTYym%2BNcQErndM%2BcZ2O8JXEC0gwRt6zW9z1h9x6LjX6ef8erVlWBM2a4dfXYANuN%2FMo5sXmnfalk99uaToCVKyGXliHmdYCNu%2FGBkOn8ldboLrlb1wgEtdGZh7JgMFoSuYf9txxshBDC4XuTDBF2tkYuQ9AzyHhBs8ZLskGM2BQX%2Fo1x13Pww6A%2FpZx%2FRa217TH9YckzFyfYIpx%2BwekOK0ggwpgyCP39SBM3F%2FzGePvX5FhKDFYeKynK9EDDl1vZzQ9ZvNXC%2FB3OPdQerocBMbOE80FwF37oSMBKkJbBlJASl1aqPp3J%2Fn%2FUiE6OzNnCC7WLZpQhqVjJVlEzjxwJQ1WtHs%2FG48ynK4EunYwaFXkIYivW773k5G4c6BQyO7CwgNA0vX9AsRGeMy%2Bv5109w%2FkGtvLmtmWmTTsK3hU3Jtd34fvnO4B%2F1ANujXh0OiSLm1OEGu3tvnBQOlI3LpiG4TdIQQs%2B%2FpCOFcDdURw%2FzMHSfiQtMW96tWRAzKR%2BOdPJ6FQuUiYr%2Fz9oXP0QER8QUsYNNlhFSYF5QNR8M%2BV%2B%2Ft4%2F5cyQZ5ZEP0KFZy3XBxsrrhvHw2lG6oE2WxqrPD8oRgCMlH08TQ909XO1SAutIOzWI%2FucqlxENzxUPexYuGiwetwIpZy9SDVgEdSj7UCnPKUhyi8ZACaZ7wc4cqQFfioVnkJ5ftlHhorngYtkQ8FFgma4x4yDv7%2FfKnX6mDWmlNLrCNbbKRURfUyLBK%2FM2COrnUphJ%2FcxP%2FRUsSf4GFLukS%2F6j86VeJv1ZakytkE0K%2FYJszfwUQq8zfKKr1rsr88mT%2BqMxL8syvF1ilki3zRx6lMn9bYmRydWvMYt%2BUfEEtzv5VgKyyf11kT28%2B3T%2F8%2FNuDt93NxUJ%2F6d%2FeuiklsTfuG6UC4cMPtCSsik1bAEtxgspwP4se1RetlNWaUymbClCBdf%2BTV4%2BZS66Z7llf0WxzdGTqAKUXzY4%2F32k15%2BU8pbOFA1SmB1ajNocFmSg1KefpvH05ObkGf%2BPO21E321ji13HgdqOiXqcH5Oq9fS6g6mbzqYp3D385sqqQb0NK6gAlV%2Fi5qqj7IdXxVEW1tbRKVRyd6UO1tEpYVAq9EhaN8gJVWSvn4wr5NrykDpCMNbblhNZqa2yVnjg6yik1tt%2FI5qt7DVyz6YpCKsyVgmgW96q%2BVk4JEfEquYaQstS2nOhacamtUhHHhzm5gPeXa5Ovswcac5ZYyr21zWVdSYm64L%2B4nr0atyNPu%2Fvu9TdwCUaD59TNN3NMoykjI1NTdKTafZMkvjq2owM1Ch9JL%2F6gqT%2Ftp8KiNtrkXxJLApfpf2p3zYEBOpUj6TMdTu3ibkm6bfG%2BmsYSr7Z7N8oF1L6a%2F1PrdWQRId9mmtQBSm6mqTniVnEyXKa3KQXRkvCZ3ENDFcTJHEZ%2FLMaVZmgU9GrLjESaQb59MqkDJOM%2BmXLiqTqLvmXhM2WfDBUNzxvvhEWDOoW%2B9dSrXTISqQYJt8akj5CMW2PKCanqFPrWhdDk6tsY%2BeQRYrZD5nTFgzqGvv3oq2PoZVIPAlLZ1UM7jqHP9iglD9oSI5PralLX39QIssr%2BzSJbnUUvU%2FbPu8G26dm%2FHWfRZ3uUyv5tiZHJFTLp62ZqhFkpgLronoze7s%2B%2F%2Fdys%2Fn0d3phTcIV%2B9FIKYxI8B7tiw9yu6fHnSNA1P2IcsPXpidL7jB6AuxGCYPurBVXKXwO4MRs4KEQBXNvk%2B87rF%2Fb6w4BfXa53fnUpVANFFm%2FYm866H7o0lXDLS2jRtpbt24Or2PsfIbbpBAQVPOm5nQ8MEdKj807xGDTnsPRwH%2F2H97FzaoURQwcQ%2Bw3GPnB5jvLIhNbWY8%2BiHQ%2Bi8uJiEO%2FSR0s8hfxthT3jcMeRxhEda3sdh5P0px03zSVzrKwcwSWZd%2Bm9Hb9khr6W7Zr0IrdnhbRkZJ%2FzE3Esrb%2BH96h7FL%2FS97vV5PKqoC1GiGS0FJ8CA896QCZkd%2F0N%3C%2Fdiagram%3E%3C%2Fmxfile%3E)

**Chapter 3 Data Definition**

**3.1 Supplier Table**

CREATE TABLE Supplier (

SupplierID CHAR(5) NOT NULL,

SupplierName VARCHAR2(30) NOT NULL,

ContactNo VARCHAR2(11) NOT NULL,

SuppEmail VARCHAR2(35) NOT NULL,

PRIMARY KEY (SupplierID),

CONSTRAINT chk\_validateEmail5 CHECK (REGEXP\_LIKE(SuppEmail,'^[a-zA-Z]\w+@(\S+)$'))

);

**3.2 Branch Table**

CREATE TABLE Branch (

BranchID CHAR(4) NOT NULL,

BranchName VARCHAR2(30) NOT NULL,

StreetAddress VARCHAR(50) NOT NULL,

PostalCode NUMBER(5) NOT NULL,

City VARCHAR(30) NOT NULL,

State CHAR(3) NOT NULL CHECK(UPPER(State) IN ('SEL','PAH','PER','TER','MAL','SAB','NSN','SAR','PEN','JOH','KED','KEL','PER','WKL','WLB','WPU')) ,

ContactNo VARCHAR2(11) NOT NULL,

PRIMARY KEY (BranchID)

);

**3.3 Employee Table**

CREATE TABLE Employee (

EmployeeID CHAR(6) NOT NULL,

Name VARCHAR2(30) NOT NULL,

BirthDate DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY') NOT NULL ,

Gender CHAR(1) NOT NULL,

ContactNo VARCHAR2(11) NOT NULL,

EmailAddr VARCHAR2(35) NOT NULL UNIQUE,

BranchID CHAR(4) NOT NULL,

JobType CHAR(2) NOT NULL,

JobID CHAR(4) NOT NULL CHECK(UPPER(JobID) IN ('MAGR', 'CASR', 'CHEF', 'SUPR', 'CLER')),

DateHired DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY') NOT NULL ,

ManagerID CHAR(6) NULL,

PRIMARY KEY (EmployeeID),

FOREIGN KEY (BranchID) REFERENCES Branch(BranchID),

FOREIGN KEY(ManagerID) REFERENCES Employee(EmployeeID),

CONSTRAINT chk\_datehired CHECK(BirthDate < DateHired),

CONSTRAINT chk\_empGender CHECK(UPPER(Gender) IN ('M','F')),

CONSTRAINT chk\_empJobtype CHECK(UPPER(JobType) IN ('PT','FT')),

CONSTRAINT chk\_validateEmail1 CHECK (REGEXP\_LIKE(EmailAddr,'^[a-zA-Z]\w+@(\S+)$'))

);

**3.4 Customer table**

CREATE TABLE Customer (

CustID CHAR(9) NOT NULL,

Name VARCHAR2(20) NOT NULL,

Gender CHAR(1) NOT NULL,

ContactNo VARCHAR2(11) NOT NULL,

EmailAddr VARCHAR2(35) NOT NULL UNIQUE,

PRIMARY KEY (CustID),

CONSTRAINT chk\_gender CHECK(Gender IN ('M','F')),

CONSTRAINT chk\_validateEmail2 CHECK (REGEXP\_LIKE(EmailAddr,'^[a-zA-Z]\w+@(\S+)$'))

);

**3.5 SaleOrder table**

CREATE TABLE SaleOrder(

SaleOrdID char(10) NOT NULL,

SaleOrdDate date DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY HH24:MI') NOT NULL ,

SaleOrdStat char(3) DEFAULT 'PEN' NOT NULL CHECK(UPPER(SaleOrdStat) IN ('PEN', 'PAC', 'COM', 'COL')) ,

BranchID char(4) NOT NULL,

OrderChannel varchar(2) DEFAULT 'DI' NOT NULL CHECK(UPPER(OrderChannel) IN ('DI','TA','DE')) ,

CustID CHAR(9) NULL,

DateTimeCompleted DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY HH24:MI') NOT NULL ,

CashierID CHAR(6) NOT NULL,

SalesTax NUMBER (5,3) NOT NULL,

GrandTotal NUMBER(8,2) NOT NULL,

PRIMARY KEY (SaleOrdID),

FOREIGN KEY (BranchID) REFERENCES Branch(BranchID),

FOREIGN KEY (CustID) REFERENCES Customer(CustID),

FOREIGN KEY (CashierID) REFERENCES Employee(EmployeeID),

CONSTRAINT chk\_SDate CHECK (DateTimeCompleted >= SaleOrdDate)

);

**3.6 CustomerFeedback table**

CREATE TABLE CustomerFeedback (

FeedbackID CHAR(10) NOT NULL,

FeedbackDate DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY') NOT NULL ,

BranchID CHAR(4) NOT NULL,

FoodRating NUMBER(1) NOT NULL,

EnvironmentRating NUMBER(1) NOT NULL,

ServiceRating NUMBER(1) NOT NULL,

PriceRating NUMBER(1) NOT NULL,

Review VARCHAR2(200),

SaleOrdID CHAR(10) NOT NULL,

CustID CHAR(9) ,

PRIMARY KEY (FeedbackID),

FOREIGN KEY(BranchID) REFERENCES Branch(BranchID),

FOREIGN KEY(CustID) REFERENCES Customer(CustID),

FOREIGN KEY(SaleOrdID) REFERENCES SaleOrder(SaleOrdID),

CONSTRAINT chk\_ratingsF CHECK (FoodRating >= 0 AND FoodRating <= 5),

CONSTRAINT chk\_ratingsE CHECK (EnvironmentRating >= 0 AND EnvironmentRating <= 5),

CONSTRAINT chk\_ratingsS CHECK (ServiceRating >= 0 AND ServiceRating <= 5),

CONSTRAINT chk\_ratingsP CHECK (PriceRating >= 0 AND PriceRating <= 5)

);

**3.7 PaymentType**

CREATE TABLE PaymentType (

PayTypeID CHAR(4) NOT NULL,

PayTypeName VARCHAR(30) not null,

PRIMARY KEY(PayTypeID)

);

**3.8 Payment table**

CREATE TABLE Payment (

PayRefNo CHAR(9) NOT NULL,

PayTypeID CHAR(4) NOT NULL,

EmployeeID CHAR(6) NOT NULL,

SaleOrdID CHAR(10) NOT NULL,

PaidDate DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY') NOT NULL ,

PaidAmt NUMBER(8, 2) NOT NULL,

PaymentStatus CHAR(1) DEFAULT 'P' NOT NULL CHECK(UPPER(PaymentStatus) IN ('P','R')) ,

PaymentChange NUMBER(6,2) DEFAULT 0 NOT NULL,

PRIMARY KEY(PayRefNo),

FOREIGN KEY(PayTypeID) REFERENCES PaymentType(PayTypeID),

FOREIGN KEY(EmployeeID) REFERENCES Employee(EmployeeID),

FOREIGN KEY(SaleOrdID) REFERENCES SaleOrder(SaleOrdID)

);

**3.9 DeliveryCompany table**

CREATE TABLE DeliveryCompany (

DeliveryCompanyID CHAR(4) NOT NULL,

DeliveryCompanyName VARCHAR(20) NOT NULL,

ContactNum VARCHAR(11) NOT NULL,

EmailAddress VARCHAR(35) NOT NULL,

PRIMARY KEY(DeliveryCompanyID),

CONSTRAINT chk\_validateEmail3 CHECK (REGEXP\_LIKE(EmailAddress,'^[a-zA-Z]\w+@(\S+)$'))

);

**3.10 DeliveryOrderInfo table**

CREATE TABLE DeliveryOrderInfo (

SaleOrdID CHAR(10) NOT NULL,

DelContactNo VARCHAR(11) NOT NULL,

DelStreetAddress VARCHAR(50) NOT NULL,

DelPostCode CHAR(5) NOT NULL,

DelCity VARCHAR(30) NOT NULL,

DelState CHAR(3) NOT NULL CHECK(UPPER(DelState) IN ('SEL','PAH','PER','TER','MAL','SAB','NSN','SAR','PEN','JOH','KED','KEL','PER','WKL','WLB','WPU')),

DeliveryCompanyID CHAR(4) NOT NULL,

ThirdPartyOrderNo CHAR(10) NOT NULL,

CollectedBy VARCHAR(40) NOT NULL,

PRIMARY KEY(SaleOrdID),

FOREIGN KEY(SaleOrdID) REFERENCES SaleOrder(SaleOrdID),

FOREIGN KEY(DeliveryCompanyID) REFERENCES DeliveryCompany(DeliveryCompanyID)

);

**3.11 Menu table**

CREATE TABLE Menu(

MenuID CHAR(4) NOT NULL,

MenuDesc VARCHAR(30) NOT NULL,

BasePrice NUMBER(5,2) NOT NULL,

MenuItemCat CHAR(2) NOT NULL CHECK(UPPER(MenuItemCat) IN ('CK','BG','BV','ST','SD')) ,

Avail\_Ind CHAR(1) DEFAULT 'A' NOT NULL CHECK(UPPER(Avail\_Ind) IN ('A','U')) ,

PRIMARY KEY (MenuID)

);

**3.12 SaleOrderItem**

CREATE TABLE SaleOrderItem(

SaleOrdID char(10) NOT NULL,

SOLineNo number(3) NOT NULL,

MenuID char(4) NOT NULL,

Qty number(3) NOT NULL,

OrderItemStatus CHAR(3) DEFAULT 'PEN' NOT NULL CHECK(UPPER(OrderItemStatus) IN ('PEN','COM')),

DeliverToTableNo number(2) NULL,

AssignedTo Char(6) NOT NULL,

DiscountAmount NUMBER(5,2) DEFAULT 0 NOT NULL,

SellingPriceBeforeDiscount NUMBER (5,2) DEFAULT 0 NOT NULL,

SellingPriceAfterDiscount NUMBER (5,2) DEFAULT 0 NOT NULL,

Subtotal NUMBER (7,2) NOT NULL,

PRIMARY KEY (SaleOrdID, SOLineNo),

FOREIGN KEY (SaleOrdID) REFERENCES SaleOrder(SaleOrdID),

FOREIGN KEY (MenuID) REFERENCES Menu(MenuID),

FOREIGN KEY (AssignedTo) REFERENCES Employee(EmployeeID),

CONSTRAINT chk\_qty CHECK (Qty > 0),

CONSTRAINT chk\_pricebeforediscount CHECK (SellingPriceBeforeDiscount > 0),

CONSTRAINT chk\_priceafterdiscount CHECK (SellingPriceAfterDiscount > 0)

);

**3.13 MenuDisc table**

CREATE TABLE MenuDisc(

MenuID CHAR(4) NOT NULL,

FromDate DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY') NOT NULL ,

ToDate DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY') NOT NULL,

DiscType CHAR(3) NOT NULL CHECK(UPPER(DiscType) IN ('PCT','CUR','QTY')),

DiscValue NUMBER(5,2) DEFAULT 0 NOT NULL,

MinOrderAmt NUMBER(5,2) DEFAULT 0,

PRIMARY KEY(MenuID, FromDate),

FOREIGN KEY (MenuID) REFERENCES Menu(MenuID),

CONSTRAINT chk\_dateMD CHECK ( ToDate >= FromDate)

);

**3.14 Ingredient table**

CREATE TABLE Ingredient(

IngID CHAR(4) NOT NULL,

IngName varchar(30) NOT NULL,

IngType CHAR(3) NOT NULL CHECK(UPPER(IngType) IN ('MEA','BUN','DAI','VEG','SAU','RIC','BEV', 'SEA', 'ICE')),

UnitOfMeasure varchar(5) DEFAULT 0 NOT NULL CHECK(UPPER(UnitOfMeasure) IN ('COUNT', 'ML', 'G', 'SLICE', 'STRIP' )),

PRIMARY KEY (IngID)

);

**3.15 Recipe table**

CREATE TABLE Recipe(

MenuID CHAR(4) NOT NULL,

IngID CHAR(4) NOT NULL,

QtyPerMenu number(6,2) NOT NULL,

PRIMARY KEY (MenuID, IngID),

FOREIGN KEY (IngID) REFERENCES Ingredient(IngID),

FOREIGN KEY (MenuID) REFERENCES Menu(MenuID),

CONSTRAINT chk\_qtypi CHECK (QtyPerMenu > 0)

);

**3.16 BranchInventory table**

CREATE TABLE BranchInventory (

IngID CHAR(4) NOT NULL,

BranchID CHAR(4) NOT NULL,

QtyOnHand Number(9,2) NOT NULL,

MinQty Number(9,2) NOT NULL,

PRIMARY KEY(IngID, BranchID),

FOREIGN KEY (IngID) REFERENCES Ingredient(IngID),

FOREIGN KEY (BranchID) REFERENCES Branch(BranchID),

CONSTRAINT chk\_qty2 CHECK (QtyOnHand >= 0),

CONSTRAINT chk\_minqty CHECK (MinQty >= 0)

);

**3.17 MenuItem table**

CREATE TABLE MenuItem(

SetID CHAR(4) NOT NULL,

MenuItemID CHAR(4) NOT NULL,

MenuItemQty NUMBER(2) NOT NULL,

PRIMARY KEY (SetID, MenuItemID),

FOREIGN KEY (SetID) REFERENCES Menu(MenuID),

FOREIGN KEY (MenuItemID) REFERENCES Menu(MenuID)

);

**3.18 PurchaseOrder table**

CREATE TABLE PurchaseOrder(

SupplierID CHAR(5) NOT NULL,

IngID CHAR(4) NOT NULL,

BranchID CHAR(4) NOT NULL,

PurchaseDate DATE DEFAULT TO\_DATE(SYSDATE, 'DD/MM/YYYY') NOT NULL,

DeliveredDate DATE NULL,

IngredientQTY NUMBER(9,2) NOT NULL,

BasePrice NUMBER(5,2) NOT NULL,

PSubtotal NUMBER(9,2) NOT NULL,

Status CHAR(2) DEFAULT 'PA' NOT NULL CHECK(UPPER(Status) IN ('PA', 'AR', 'PD')),

PRIMARY KEY(PurchaseDate, SupplierID, BranchID, IngID),

FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),

FOREIGN KEY (IngID) REFERENCES Ingredient,

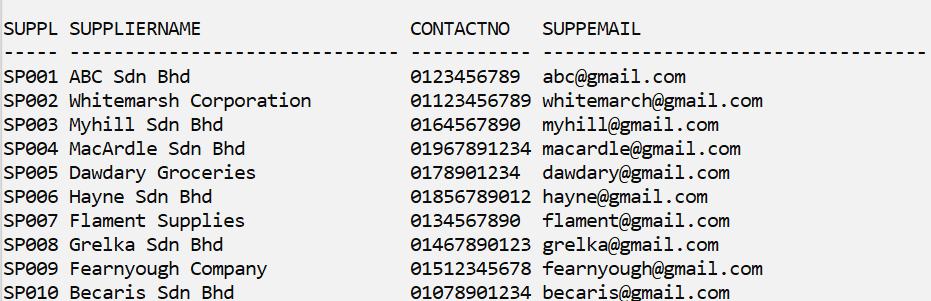
FOREIGN KEY (BranchID) REFERENCES Branch,

CONSTRAINT chk\_datePO CHECK (DeliveredDate >= PurchaseDate)

);

**3.2 (10 Sample records for each table)**

**1.Supplier table**

****

**2.Branch table**

****

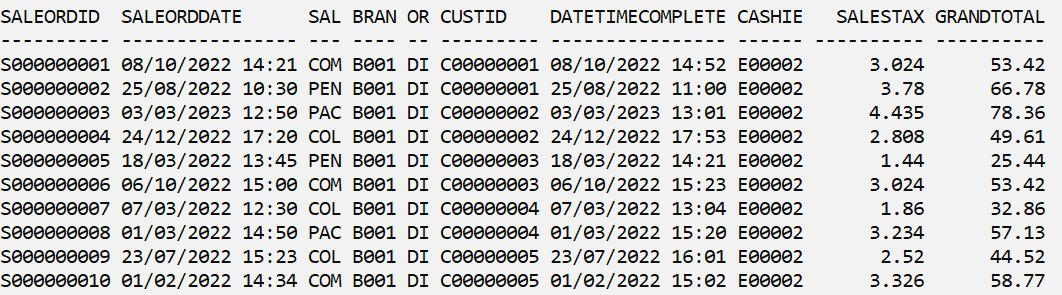
**3.Employee table**

****

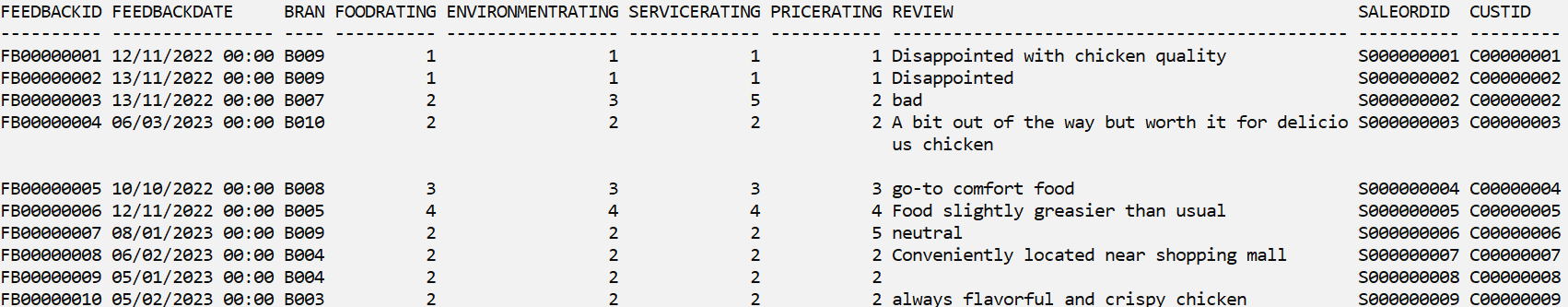
**4.Customer table**

****

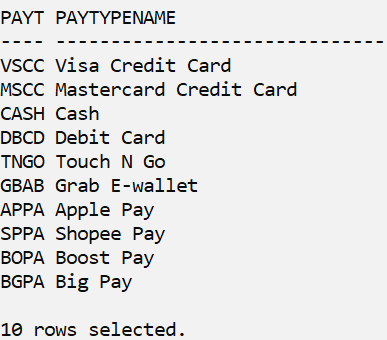
**5.Sale Order table**

****

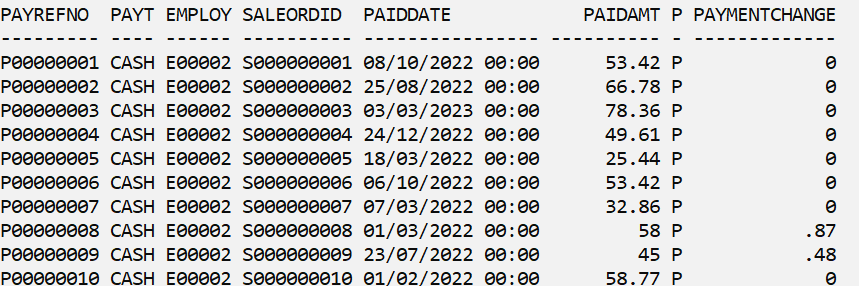
**6.Customer feedback table**

****

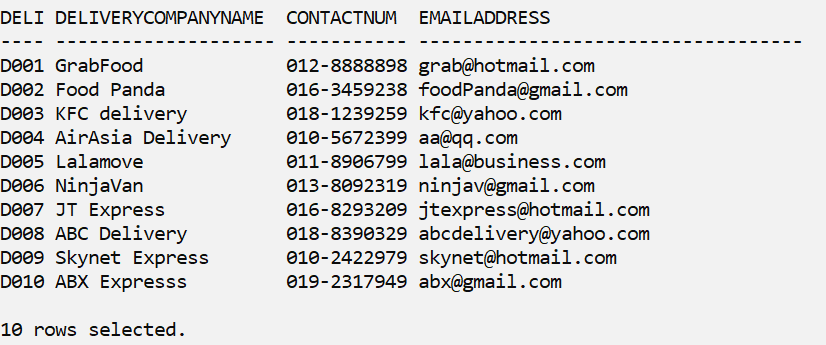
**7.Payment Type table**

****

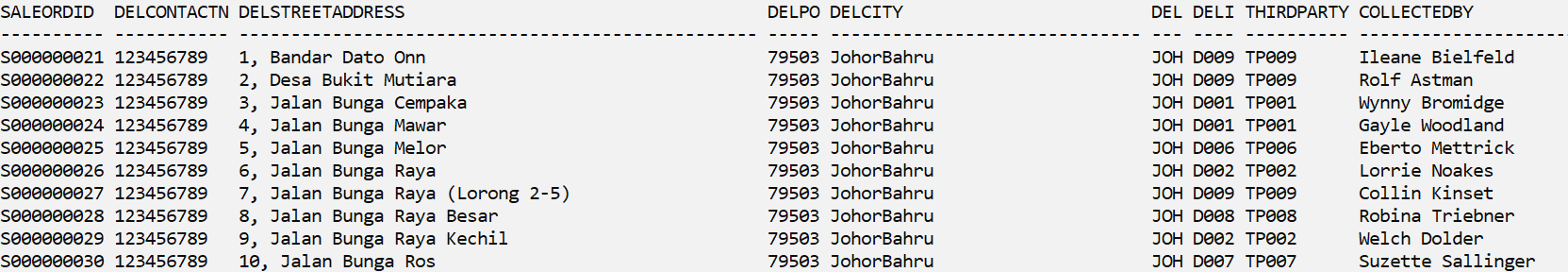
**8. Payment table**

****

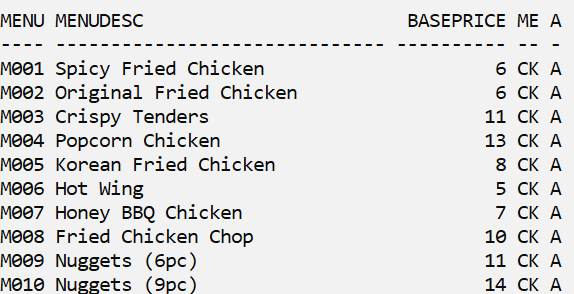
**9. Delivery Company Table**

****

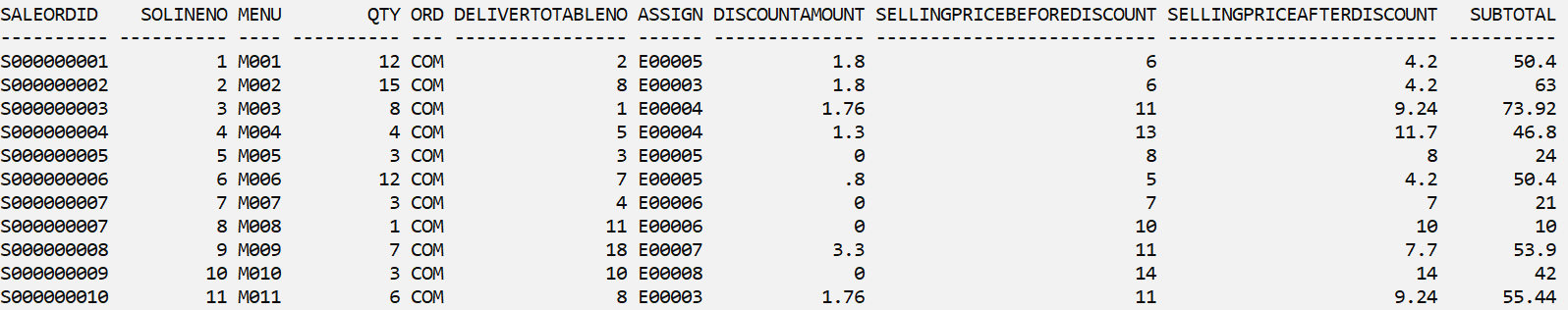
**10. Delivery Order Info table**

****

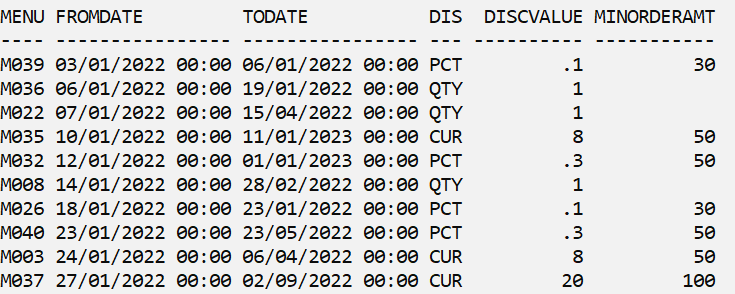
**11. Menu table**

****

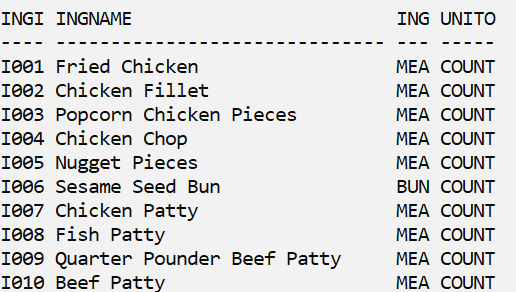
**12. Sale order item table**

****

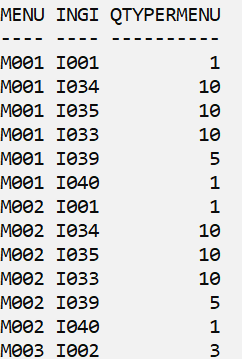
**13. Menu Disc table**

****

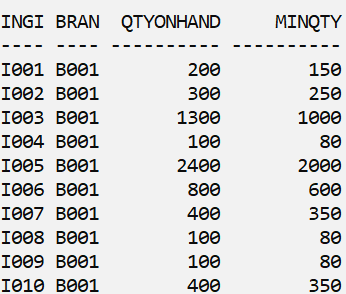
**14. Ingredient table**

****

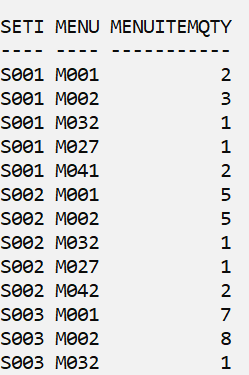
**15. Recipe table**

****

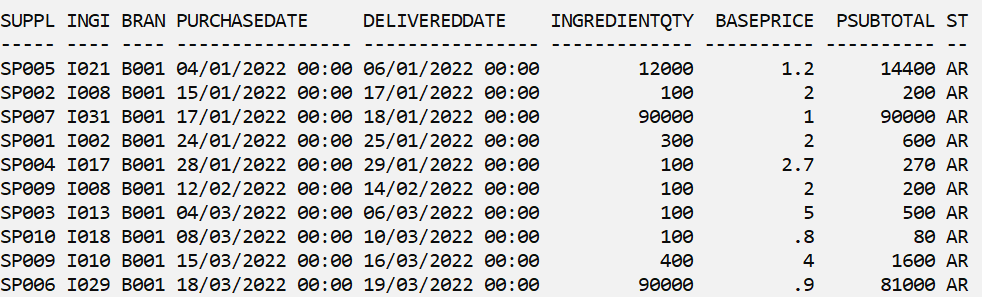
**16. Branch inventory table**

****

**17. Menu item table**

****

**18. Purchase order table**

****

**Chapter 4 Queries, Procedures, Triggers and Reports**

**4.1 (Ong Weng Kai)**

**4.1.1 Query 1:** This query is to display Average food, environment, service and price rating in each branch.

**Purpose:**

* **Customer Demographics:**

It can know the total number of male and female respondents for each branch to know the customer demographic so that it can help the company tailor its marketing campaign.

* **Feedback Analysis:**

This query allows the management to know which branch in what area is performing well and where has room for improvement. Management can get insight through the BranchID and average food, environment, and service ratings to know which branch outperforms. If a branch has an excellent rating, we can give them some reward or bonus to encourage them to provide a better service. For branches with a low service rating, the company can send someone to do an investigation or train the employees to serve customers better.

* **Strategic Decision**

This query can also guide a strategic decision if a branch has a low environmental rating; it might indicate that the company can change the ambience or renovate to keep and attract customers.

**SQL statement:**

SET linesize 1000

CREATE VIEW CustomerSummary AS

SELECT

Branch.BranchID,

SUM(CASE WHEN Customer.Gender = 'M' THEN 1 ELSE 0 END) AS Total\_Male\_Cust,

SUM(CASE WHEN Customer.Gender = 'F' THEN 1 ELSE 0 END) AS Total\_Female\_Cust,

ROUND(AVG(CustomerFeedback.FoodRating), 2) AS Food\_Rating,

ROUND(AVG(CustomerFeedback.EnvironmentRating), 2) AS Environment\_Rating,

ROUND(AVG(CustomerFeedback.ServiceRating), 2) AS Service\_Rating,

ROUND(AVG(CustomerFeedback.PriceRating), 2) AS Price\_Rating

FROM

Branch

JOIN CustomerFeedback ON Branch.BranchID = CustomerFeedback.BranchID JOIN Customer ON CustomerFeedback.CustID = Customer.CustID

GROUP BY

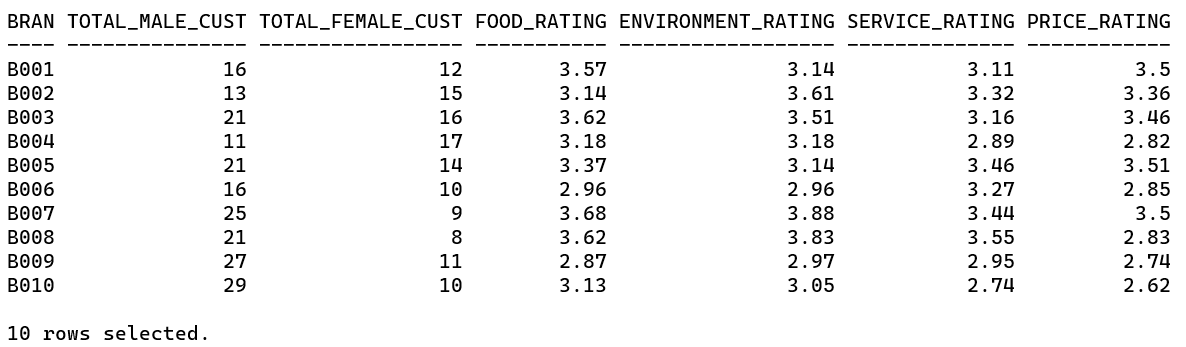
Branch.BranchID;

SET LINESIZE 120

SELECT \* FROM CustomerSummary ORDER BY BranchID ASC;

DROP VIEW CustomerSummary;

**Sample Output:**



**4.1.2 Query 2:** The query retrieves the average unit price of a specific ingredient from different suppliers, along with the supplier's name and contact number.

**Purpose :**

The query retrieves the average unit price of a specific ingredient from different suppliers and the supplier's name and contact number.

First is saving cost, as being able to identify the average unit price from a different supplier, management can choose the most cost-effective ingredient, especially when the ingredient is purchased in large quantities.

Which suppliers offer the most cost-effective rates for specific ingredients? Businesses can leverage and negotiate better deals with the suppliers to get better deals with suppliers. This query also ensures that the company can quickly source the ingredient, especially if that ingredient is essential for the business operation. Even though this query mainly focuses on price, if one supplier has a higher price because of better quality, it also indicates that the company can consider buying from that supplier if they want to prepare for the high-end product.

**SQL statement:**

ACCEPT IngredientCode CHAR PROMPT 'Please enter Ingredient Code > '

SELECT

Supplier.SupplierName,

Supplier.ContactNo,

AVG(PurchaseOrder.Psubtotal / PurchaseOrder.IngredientQty) AS Avg\_Unit\_Price,

Ingredient.IngName AS IngredientName

FROM

Supplier

JOIN PurchaseOrder ON Supplier.SupplierID = PurchaseOrder.SupplierID

JOIN Ingredient ON Ingredient.IngID = PurchaseOrder.IngID

WHERE

Ingredient.IngID = '&IngredientCode'

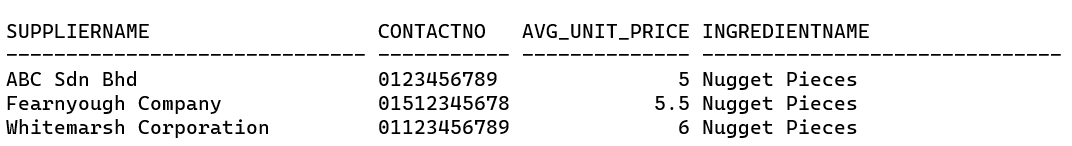
GROUP BY

Supplier.SupplierName,

Supplier.ContactNo,

Ingredient.IngName;

**Sample output:**

****

**Tips**

Insert format I00X

**4.1.3 Trigger 1: Low rating feedback alert system**

**Purpose:**

Providing immediate awareness of negative feedback is essential for fast-food restaurants because addressing the issues can improve customer satisfaction. This program allows the staff to take action immediately after receiving feedback. By alerting low ratings, management can quickly address and identify the quality control issues to ensure consistent service. Maintaining a good reputation is vital in the age of social media and online. It addresses the negative feedback right after it could win back dissatisfied customers or even prevent broader negative publicity. This program also allows the business to enhance the customer experience by demonstrating that the restaurant values customer feedback.

**SQL Statement:**

CREATE OR REPLACE TRIGGER trg\_notify\_bad\_feedback

AFTER INSERT OR UPDATE ON CustomerFeedback

FOR EACH ROW

BEGIN

-- Check if any of the ratings is below 2.5

IF :NEW.FoodRating < 2.5 OR

:NEW.EnvironmentRating < 2.5 OR

:NEW.ServiceRating < 2.5 OR

:NEW.PriceRating < 2.5 THEN

DBMS\_OUTPUT.PUT\_LINE('================================================');

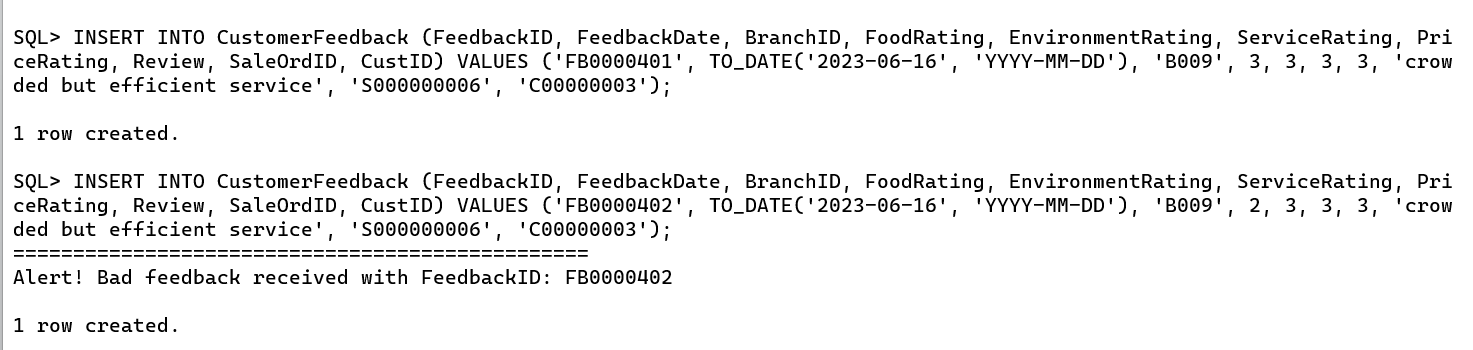
DBMS\_OUTPUT.PUT\_LINE('Alert! Bad feedback received with FeedbackID: ' || :NEW.FeedbackID);

END IF;

END;

/

**Sample Output:**

****

***Command***

INSERT INTO CustomerFeedback (FeedbackID, FeedbackDate, BranchID, FoodRating, EnvironmentRating, ServiceRating, PriceRating, Review, SaleOrdID, CustID) VALUES ('FB0000403', TO\_DATE('2023-06-16', 'YYYY-MM-DD'), 'B009', 2, 3, 3, 3, 'crowded but efficient service', 'S000000006', 'C00000003');

FB0000400 onward

**4.1.4 Trigger2 : Check whether new menu item have discount.**

**Purpose:**

Check if the new menu item added to the menu table has an associated promotional discount in the Menu Disc table; if not, output a message suggesting adding a promotional deal. This trigger checks whether the new item lacks a promotional value; this is especially useful as promotional discounts can be one of the most significant drivers for sales especially for new menu items. By alerting the management, they can consider adding one to attract customers in case management forgets about it. This program will alert every insert operation on the menu table If the discount is missing. Offering discounts on new menu items would encourage more customers to try them. More Discounts can lead to more feedback and reviews, And this is very valuable as this allows businesses to make any necessary improvement, for example, slightly changing the flavour. In addition, if the new menu item involves ingredients not commonly used in other dishes, this can help increase the turnover rate of that particular item. And promoting more people to try new menus allows the business to know the customer preferences more easily.

**Sql Code:**

CREATE OR REPLACE TRIGGER trg\_new\_menu\_item

AFTER INSERT ON Menu

FOR EACH ROW

DECLARE

v\_discount\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_discount\_count

FROM MenuDisc

WHERE MenuID = :NEW.MenuID;

IF v\_discount\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('============================================================= ');

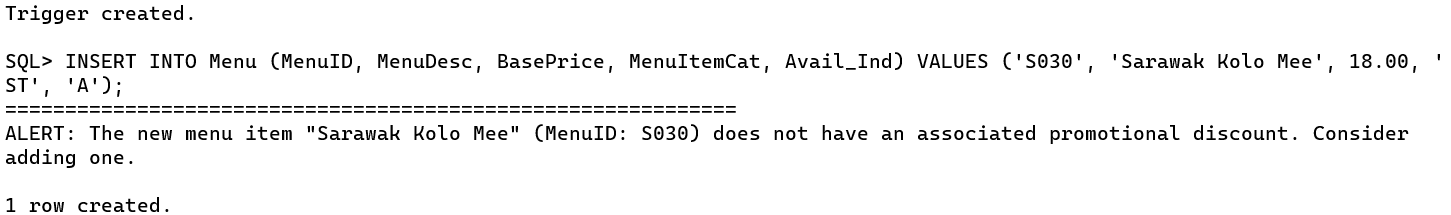
DBMS\_OUTPUT.PUT\_LINE('ALERT: The new menu item "' || :NEW.MenuDesc || '" (MenuID: ' || :NEW.MenuID || ') does not have an associated promotional discount. Consider adding one.');

END IF;

END trg\_new\_menu\_item;

/

**Sample Output:**

****

**Command**

INSERT INTO Menu (MenuID, MenuDesc, BasePrice, MenuItemCat, Avail\_Ind) VALUES ('S031', 'Sarawak Kolo Mee', 18.00, 'ST', 'A');

MenuID can used start from : S030

or

MenuID can used start from : M050

**4.1.3 Procedure 1: Fetch all available discount at current time**

**Purpose:**

This procedure fetches all active discounts available for menu items. So, staff can inform customers about ongoing discounts by upselling or recommending dishes that have active promotions. For fast food restaurants with multiple branches, this procedure ensures that all branches have consistent information about ongoing promotions, providing a uniform customer experience so the company can operate efficiently. Management can be aware of ongoing promotions to ensure they align with the restaurant's marketing and sales strategies. This procedure can help the inventory team and kitchen stock the items and ingredients at a significant discount as, Generally, the thing at a deal would result in higher sales. After that, a value is a cost for the company, so fetching an active discount allows the finance team to make accurate revenue projections. After that, the marketing team can use the information to design targeted advertisements, which can be highlighted in marketing campaigns to attract more customers. For restaurants with multiple branches, this procedure ensures that all branches have consistent information about ongoing promotions, providing a uniform customer experience so the company can operate efficiently.

**SQL code:**

SET linesize 1000

CREATE OR REPLACE PROCEDURE FetchActiveDiscounts IS

CURSOR c\_discounts IS

SELECT

m.MenuID,

m.MenuDesc,

d.DiscType,

d.DiscValue,

d.ToDate, -- Moved this line up

d.MinOrderAmt

FROM MenuDisc d

JOIN Menu m ON m.MenuID = d.MenuID

WHERE SYSDATE BETWEEN d.FromDate AND d.ToDate;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE(RPAD('MenuID', 10) || RPAD('MenuDesc', 30) || RPAD('DiscType', 15) || RPAD('DiscValue', 15) || RPAD('ToDate', 15) || 'MinOrderAmt');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

FOR r\_discount IN c\_discounts LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(r\_discount.MenuID, 10) ||

RPAD(SUBSTR(r\_discount.MenuDesc, 1, 30), 30) ||

RPAD(r\_discount.DiscType, 15) ||

RPAD(r\_discount.DiscValue, 15) ||

RPAD(TO\_CHAR(r\_discount.ToDate, 'DD/MM/YYYY'), 15) || -- Added RPAD here

r\_discount.MinOrderAmt

);

END LOOP;

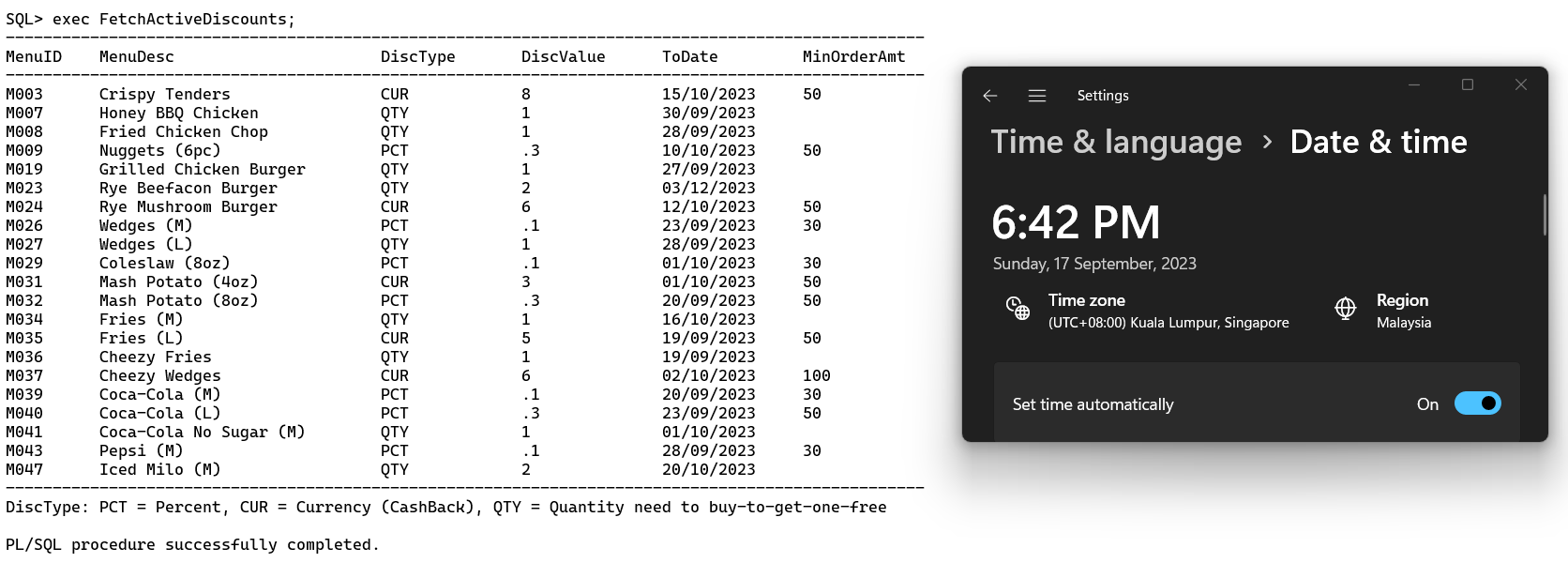
DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('DiscType: PCT = Percent, CUR = Currency (CashBack), QTY = Quantity need to buy-to-get-one-free');

END FetchActiveDiscounts;

/

**Sample output:**



**Remarks:** No ToDate is late than current Date :17 September 2023

**Command**

exec FetchActiveDiscounts;

**Procedure2:** Fetch all customerfeedback remark based on specific rating

**Purpose:**

This procedure allows the user to retrieve based on a specific rating quickly. It can fetch all the ratings related to the particular area, so this helps in understanding regions that need improvement. By categorizing feedback into ‘Food’, ‘Environment’, ‘Service’ and ‘Price’, the restaurant can know precisely where it needs to be improved. For example, if we receive a star comment regarding ‘Service’, indicate that the staff might need service training for that specific branch. This also allows the restaurant to tailor more promotion and marketing; for example, if receiving a low rating for ‘Price’, They might consider doing special deals, promotions or lowering their price to attract more customers. If they receive a high rating for food, indicate that they can use this as a marketing point. For a fast-food restaurant with multiple branches, the feedback might vary by branch. By retrieving the feedback by rating, the restaurant can know which branches are performing well and which ones need more attention. For fast food restaurants, swift response to negative feedback is essential. If one restaurant is receiving a low rating, it can take action to address this problem.

**Sql code:**

SET linesize 1000

CREATE OR REPLACE PROCEDURE FeedbackByRating(

p\_category IN VARCHAR2,

p\_exact\_rating IN NUMBER

) IS

BEGIN

-- Header

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE(RPAD('BranchID', 10) || RPAD('FeedbackID', 15) || RPAD('Rating', 10) || 'Review');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

-- Check which category is selected and fetch feedback based on the exact rating

CASE p\_category

WHEN 'Food' THEN

FOR rec IN (SELECT \* FROM CustomerFeedback

WHERE FoodRating = p\_exact\_rating

ORDER BY BranchID) LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(rec.BranchID, 10) ||

RPAD(rec.FeedbackID, 15) ||

RPAD(rec.FoodRating, 10) ||

rec.Review

);

END LOOP;

WHEN 'Environment' THEN

FOR rec IN (SELECT \* FROM CustomerFeedback

WHERE EnvironmentRating = p\_exact\_rating

ORDER BY BranchID) LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(rec.BranchID, 10) ||

RPAD(rec.FeedbackID, 15) ||

RPAD(rec.EnvironmentRating, 10) ||

rec.Review

);

END LOOP;

WHEN 'Service' THEN

FOR rec IN (SELECT \* FROM CustomerFeedback

WHERE ServiceRating = p\_exact\_rating

ORDER BY BranchID) LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(rec.BranchID, 10) ||

RPAD(rec.FeedbackID, 15) ||

RPAD(rec.ServiceRating, 10) ||

rec.Review

);

END LOOP;

WHEN 'Price' THEN

FOR rec IN (SELECT \* FROM CustomerFeedback

WHERE PriceRating = p\_exact\_rating

ORDER BY BranchID) LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(rec.BranchID, 10) ||

RPAD(rec.FeedbackID, 15) ||

RPAD(rec.PriceRating, 10) ||

rec.Review

);

END LOOP;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Invalid category provided.');

END CASE;

-- Footer

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

EXCEPTION

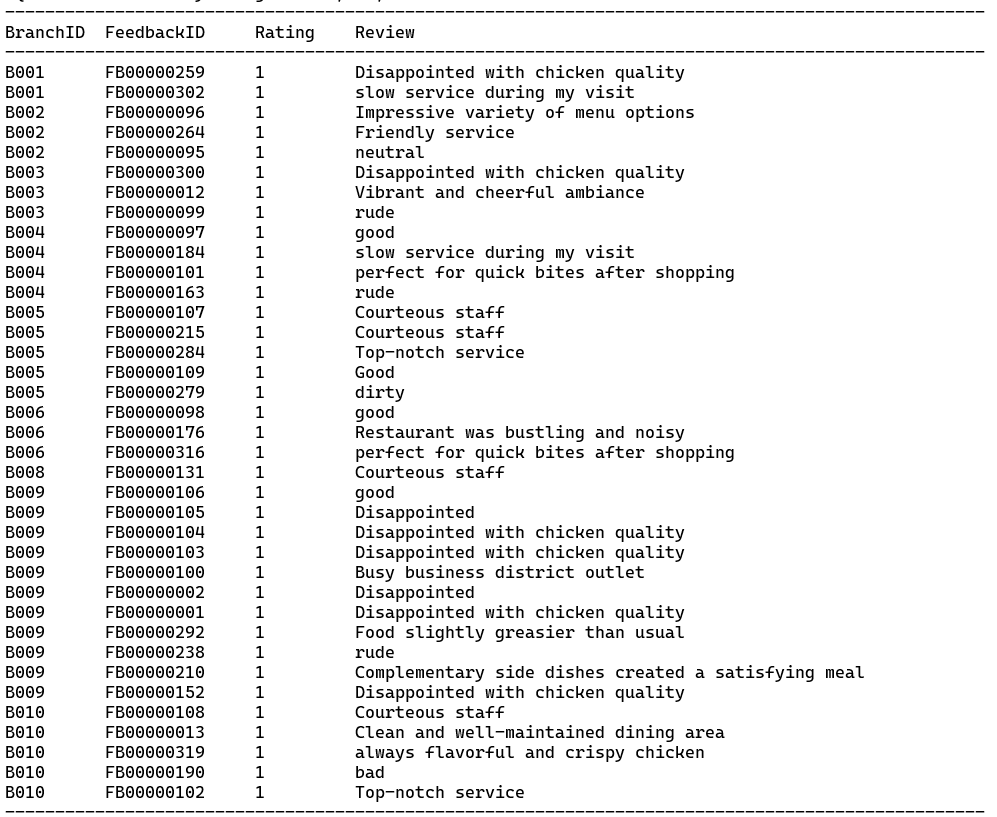
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END FeedbackByRating;

/

**Sample Output:**

****

**Command:**

exec FeedbackByRating('Food', 1);

exec FeedbackByRating('Service', 1);

exec FeedbackByRating('Environment', 1);

exec FeedbackByRating('Price', 1);

**Tips**

The word in red colour can be 1,2,3,4,5

**Report 1 :** Fetch customer feedback rating in 4 quarters (1 year) and display the branch, total customers give a rating.

**Purpose:**

This report allows the management to monitor the performance of each branch every quarter. So the company can identify the trends, strengths and areas for improvement. Most importantly, it fosters a culture of continuous improvement within the organization. By reviewing performance data, the company can implement changes and innovations to enhance customer satisfaction and profitability. The company can know customer satisfaction every quarter; they can understand whether the company is performing better each quarter or whether the service has worsened over time. Companies can reward branches that serve well, and for branches that perform poorly, they can give some punishment or training to improve the branch quality. The company can also choose one branch with a high rating for its service and decide to highlight this in marketing campaigns. The company can make an informed decision about resource allocation or expansion plans by reviewing the customer satisfaction of each branch.

**SQL Code:**

CREATE OR REPLACE PROCEDURE GenQuarterlyReport IS

-- Variables to hold data for each branch and quarter

CurrentBranchID branch.BranchID%TYPE;

QuarterStart DATE := TO\_DATE('08/04/2022', 'DD/MM/YYYY');

QuarterEnd DATE;

AvgFoodRating NUMBER(5,2);

AvgEnvironmentRating NUMBER(5,2);

AvgServiceRating NUMBER(5,2);

AvgPriceRating NUMBER(5,2);

TotalCustomers NUMBER;

-- Cursor to loop through each branch

CURSOR cur IS SELECT BranchID FROM Branch;

BEGIN

-- Loop through the branches using the cursor

FOR rec IN cur LOOP

CurrentBranchID := rec.BranchID;

-- Display header for each branch

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Branch ID: ' || CurrentBranchID);

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------');

-- Reset the QuarterStart for each branch

QuarterStart := TO\_DATE('08/04/2022', 'DD/MM/YYYY');

-- Loop through each quarter

WHILE QuarterStart < TO\_DATE('08/04/2023', 'DD/MM/YYYY') LOOP

QuarterEnd := ADD\_MONTHS(QuarterStart, 3);

-- Calculate average ratings and total customers for the current branch and quarter

SELECT

ROUND(AVG(FoodRating), 2),

ROUND(AVG(EnvironmentRating), 2),

ROUND(AVG(ServiceRating), 2),

ROUND(AVG(PriceRating), 2),

COUNT(DISTINCT CustID)

INTO

AvgFoodRating,

AvgEnvironmentRating,

AvgServiceRating,

AvgPriceRating,

TotalCustomers

FROM

CustomerFeedback

WHERE

BranchID = CurrentBranchID AND

FeedbackDate BETWEEN QuarterStart AND QuarterEnd;

-- Output the data for each quarter

DBMS\_OUTPUT.PUT\_LINE('Quarter Start: ' || TO\_CHAR(QuarterStart, 'DD/MM/YYYY'));

DBMS\_OUTPUT.PUT\_LINE('Food Rating : ' || AvgFoodRating);

DBMS\_OUTPUT.PUT\_LINE('Environment Rating: ' || AvgEnvironmentRating);

DBMS\_OUTPUT.PUT\_LINE('Service Rating : ' || AvgServiceRating);

DBMS\_OUTPUT.PUT\_LINE('Price Rating : ' || AvgPriceRating);

DBMS\_OUTPUT.PUT\_LINE('Total Customers : ' || TotalCustomers);

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------');

-- Move to the next quarter

QuarterStart := QuarterEnd;

END LOOP;

END LOOP;

EXCEPTION

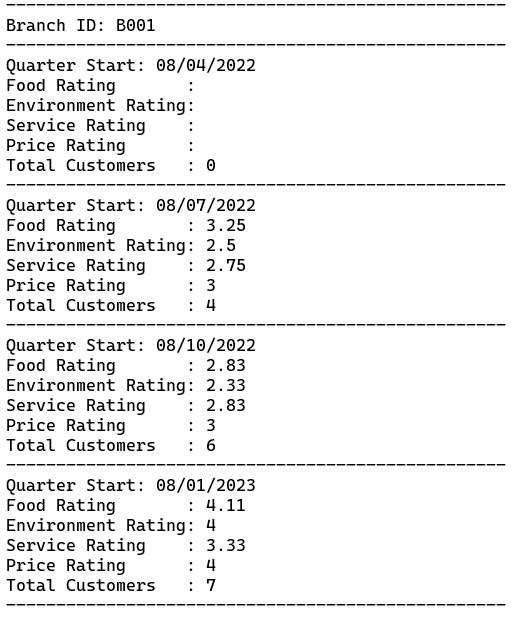
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

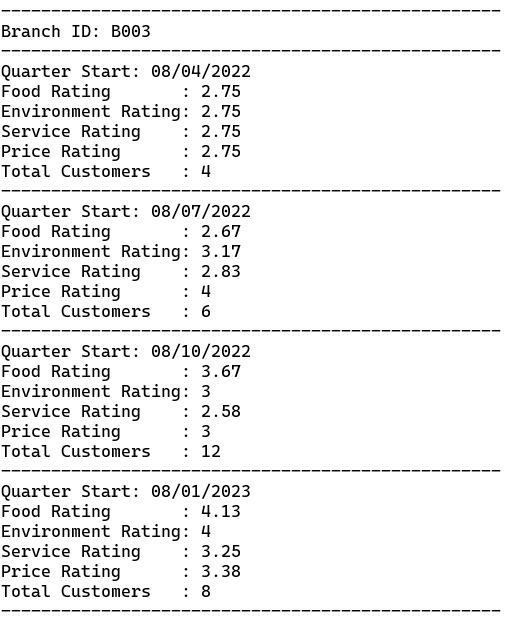
END GenQuarterlyReport;

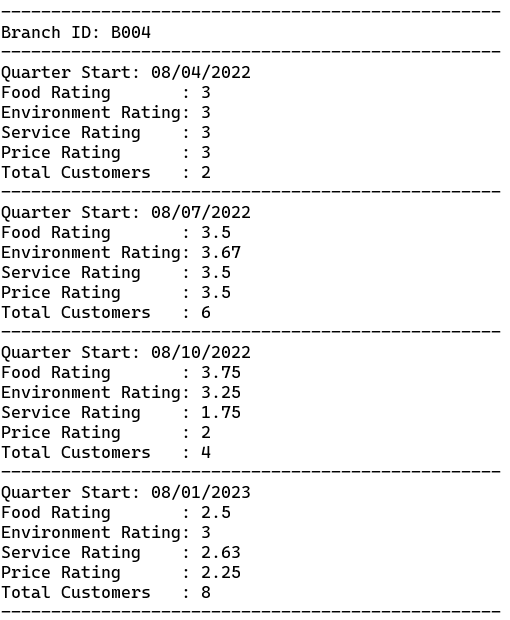
/

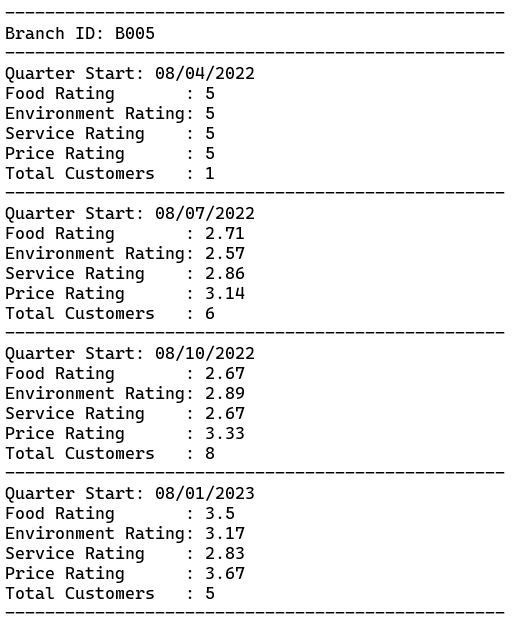
**Sample output:**

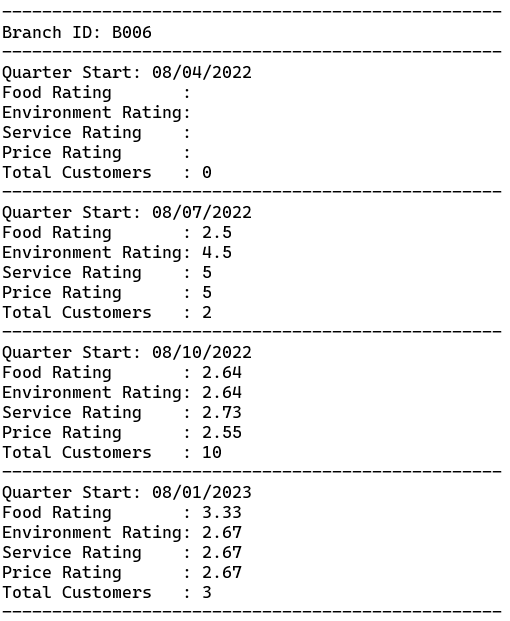


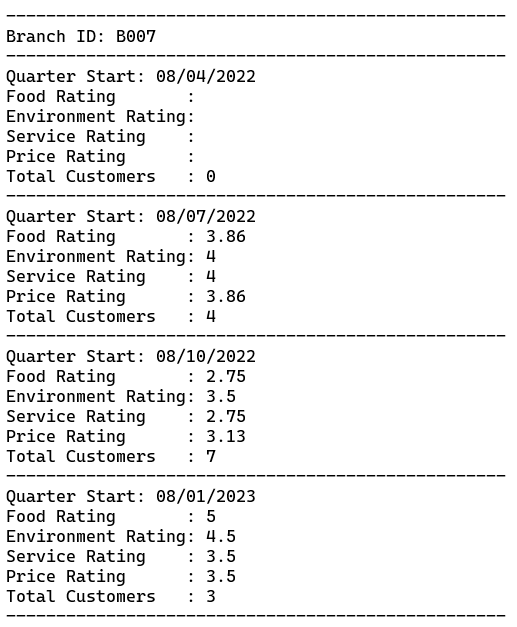
****

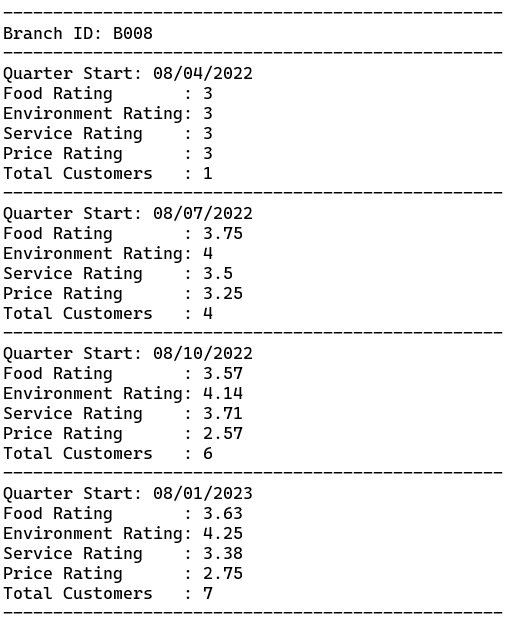
****

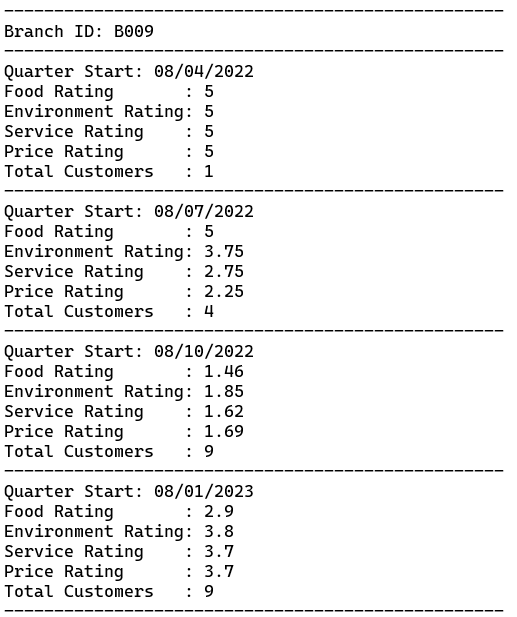
****

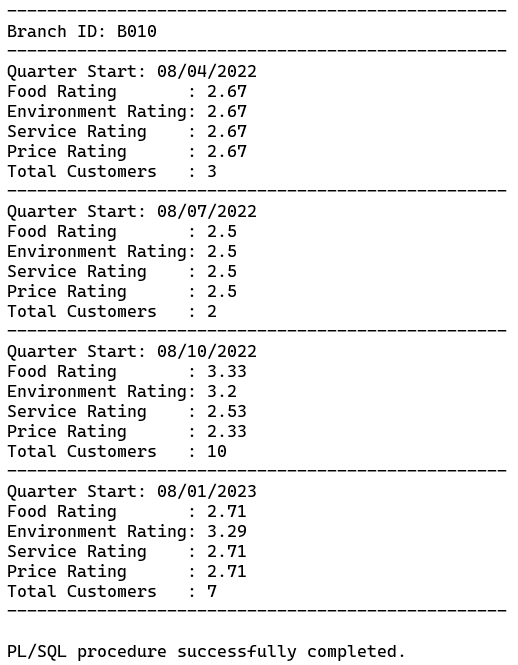
****

****

****

****

****

****

**Command**

EXEC GenQuarterlyReport;

**Report 2:** Employee Report.

**Purpose:**

The report can make sure the gender of the company is equally distributed to ensure inclusivity and diversity in hiring practice; this can provide a more diverse perspective and eventually contribute to business growth. The balanced gender report can be used as part of the company's Corporate Social Responsibility or Brand Image, showing commitment to equality, and this can create a positive image and attract more customers who favour the company to be transparent and promote fairness. After that, knowing how many employees are full-time and part-time allows management to effectively allocate resources during peak and off-peak hours to ensure the company operate effectively. After that, knowing the full-time and part-time employee counts is crucial, especially if a significant event is coming up; knowing the available workforce ensures the customer will have a great experience. Understanding the many job roles in a company enables the administration to design a training program to improve its quality. For example, if too many employees are in one part or fewer in another, it indicates the need for cross-training or hiring in specific areas. Not only that, if a restaurant plans to open a new branch, knowing the current numbers can help make budgetary decisions. Knowing employee distribution would eventually increase customer satisfaction. If there is a sufficient number of chefs, cashiers, or any other vital roles, it would lead to long waiting times or a drop in service quality. Lastly, each position has a different pay scale; by knowing the job distribution, the team can forecast monthly salaries accurately.

**SQL Code:**

CREATE OR REPLACE PROCEDURE EmployeeReport IS

v\_genderDesc VARCHAR2(10);

v\_count NUMBER;

v\_jobTitle VARCHAR2(20);

v\_jobTypeDesc VARCHAR2(20);

CURSOR gender\_cursor IS

SELECT

CASE

WHEN Gender = 'M' THEN 'Male'

WHEN Gender = 'F' THEN 'Female'

END AS GenderDesc,

COUNT(\*) as TotalCount

FROM Employee

GROUP BY Gender;

CURSOR jobType\_cursor IS

SELECT

CASE

WHEN JobType = 'PT' THEN 'PartTime'

WHEN JobType = 'FT' THEN 'FullTime'

END AS JobTypeDesc,

COUNT(\*) as TotalCount

FROM Employee

GROUP BY JobType;

CURSOR jobID\_cursor IS

SELECT

CASE

WHEN JobID = 'SUPR' THEN 'Supervisor'

WHEN JobID = 'CLER' THEN 'Cleaner'

WHEN JobID = 'MAGR' THEN 'Manager'

WHEN JobID = 'CASR' THEN 'Cashier'

WHEN JobID = 'CHEF' THEN 'Chef'

END AS JobTitleDesc,

COUNT(\*) as TotalCount

FROM Employee

GROUP BY JobID;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Count of Male and Female Employees:');

DBMS\_OUTPUT.PUT\_LINE('----------------------------------');

OPEN gender\_cursor;

LOOP

FETCH gender\_cursor INTO v\_genderDesc, v\_count;

EXIT WHEN gender\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Gender: ' || v\_genderDesc || ' Count: ' || v\_count);

END LOOP;

CLOSE gender\_cursor;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Count of Employees for Each JobType:');

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

OPEN jobType\_cursor;

LOOP

FETCH jobType\_cursor INTO v\_jobTypeDesc, v\_count;

EXIT WHEN jobType\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('JobType: ' || v\_jobTypeDesc || ' Count: ' || v\_count);

END LOOP;

CLOSE jobType\_cursor;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Count of Employees for Each JobID:');

DBMS\_OUTPUT.PUT\_LINE('---------------------------------');

OPEN jobID\_cursor;

LOOP

FETCH jobID\_cursor INTO v\_jobTitle, v\_count;

EXIT WHEN jobID\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('JobID: ' || v\_jobTitle || ' Count: ' || v\_count);

END LOOP;

CLOSE jobID\_cursor;

EXCEPTION

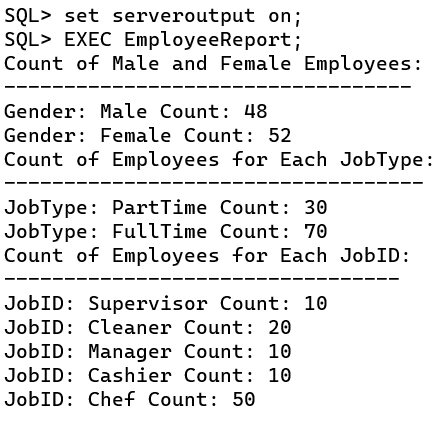
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END EmployeeReport;

/

**Sample output:**

****

**Command**

EXEC EmployeeReport;

**4.2 (Ryan Kho Yuen Thian)**

**4.2.1 Query 1: For a given date range, list the Average sales revenues in descending order by Day of the Week (for eg Monday, Tuesday, Wednesday, etc) for each Branch**

**Purpose:** This is a management report which displays the Sales Revenues for each Branch by Day of the Week (Monday, …, Sunday) for a selected period. For a given Branch, the sales revenues are displayed in Descending order i.e. highest sales to lowest sales (not by chronological order). Since the sales amount provides a measure of how busy a branch is on a particular day of the week, this info is useful to the management in allocating different numbers of workers to the branch for the different days of the week e.g. more workers on weekends than weekdays. This query should fall under the ‘Operational’ category since it involves the daily operations of the branches.

**SQL statement:**

cl scr;

alter session set nls\_date\_format = 'DD/MM/YYYY';

SET PAGESIZE 60

SET LINESIZE 100

PROMPT Please enter the following details

ACCEPT from\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter FROM Date > '

ACCEPT thru\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter TO Date > '

TTITLE CENTER "The Fast Food Restaurant" SKIP 2 -

LEFT "Mkt Department" -

RIGHT "Branch Average Sales Revenues (Highest to Lowest) by Day of Week" SKIP 1 -

LEFT "=====================================================" -

"=================================================="

BTITLE LEFT "=====================================================" -

"==================================================" -

SKIP 1 -

RIGHT "Page " FORMAT 999 SQL.PNO

COLUMN BranchNo Heading "Branch No" FORMAT A9

COLUMN BranchName Heading "Branch Name"

COLUMN DayOfWeek HEADING "Day"

COLUMN BrnDowAvg HEADING "Average Sales (RM)" FORMAT 999,999.99

BREAK ON BranchNo SKIP 1 NODUPLICATES -

ON BranchName SKIP 1 NODUPLICATES -

SELECT BranchNo, BranchName, DayOfWeek, AVG(BrnOrdDtTotal) AS BrnDowAvg

FROM

(SELECT SaleOrder.BranchID AS BranchNo, BranchName, SaleOrdDate, SUM(GrandTotal) AS BrnOrdDtTotal, TO\_CHAR(SaleOrdDate, 'DAY') AS DayOfWeek

FROM SaleOrder INNER JOIN Branch ON SaleOrder.BranchID = Branch.BranchID

WHERE SaleOrdStat IN ('COM', 'COL') AND SaleOrdDate BETWEEN TO\_DATE('&&from\_date','DD/MM/YYYY') AND TO\_DATE('&&thru\_date','DD/MM/YYYY')

GROUP BY SaleOrder.BranchID, BranchName, SaleOrdDate)

GROUP BY BranchNo, BranchName, DayOfWeek

ORDER BY BranchNo, BranchName, BrnDowAvg desc;

clear columns

clear breaks

clear computes

ttitle off

btitle off

**Sample Output:**









**4.2.2 Query 2: List all Ala Carte menu items which currently have discounts grouped by Menu Item Category**

**Purpose:** This query lists out the Ala Carte menu items which are on promotion during a selected period (specified by the user in the From Date and To Date prompts). The menu items are grouped by categories (Beverages, Burgers, Salads, Chicken, etc.). For each menu item, the type of promotion is also displayed. There are 3 types of promotion. The DiscountType ‘PCT’ means Percentage’. For this discount type, the DiscountValue field will contain the proportion e.g. for 20% discount, it contains 0.20. The ‘CUR’ discount type means Currency. For this discount type, the DiscountValue field will contain the special price (in RM) that the menu item is offered at. If the discount type is ‘QTY’, it means Quantity. How this discount type works is like this: if DiscountValue contains e.g. 3, this means the customer has to buy 3 of the menu item in order to get 1 free. The MinOrderAmt is the Minimum Order Amount that the customer must meet in order to qualify for the discount. If MinOrderAmt is 0, it means there is no minimum order amount. Note that this SELECT query can easily be altered to produce a list of ‘Set Meals’ which are on promotion. This is done by simply changing the NOT IN used in the subquery to NOT. The promotion info generated from this query can be made available to the public (e.g. in promotions page on the company’s website) as well as to the management. As promotions are short-term efforts to boost sales, this query should fall under the ‘Tactical’ category.

**SQL statement:**

cl scr;

SET PAGESIZE 30

SET LINESIZE 190

alter session set nls\_date\_format = 'DD/MM/YYYY';

TTITLE CENTER "The Fast Food Restaurant" SKIP 2 -

CENTER "Ala Carte Products Currently on Promotion" SKIP 2 -

LEFT "LEGEND: CK = Chicken, BG = Burger, BV = Beverage, SD = Side Dish, ST = SET" SKIP 1 -

LEFT "==============================================================================================" -

"==========================================================================" -

BTITLE LEFT "==============================================================================================" -

"==========================================================================" -

CENTER "(Page " FORMAT 999 SQL.PNO ")"

COLUMN MenuItemCat HEADING "Category" FORMAT A8

COLUMN MenuID HEADING "Product ID" FORMAT A10

COLUMN MenuDesc HEADING "Product Name"

COLUMN BasePrice HEADING "Normal Price" FORMAT 999.99

COLUMN DiscountType HEADING "Discount Type"

COLUMN DiscountedPrice HEADING "Discounted Price" FORMAT 999.99

COLUMN DiscountPercent HEADING "Discount %" FORMAT 999.99

COLUMN NumberToBuy HEADING "No. to Buy (Free 1)" FORMAT 99

COLUMN MinOrderAmt HEADING "Min. Order Amt" FORMAT 999.99

COLUMN FromDate HEADING "Start Date"

COLUMN ToDate HEADING "End Date"

BREAK ON MenuItemCat SKIP 1 NODUPLICATES -

SELECT MenuItemCat, M.MenuID, MenuDesc, BasePrice,

CASE DiscType

WHEN 'CUR' THEN 'Special Price'

WHEN 'PCT' THEN 'Percentage Discount'

WHEN 'QTY' THEN 'Buy X Get 1 Free'

END AS DiscountType,

CASE DiscType

WHEN 'CUR' THEN DiscValue

WHEN 'PCT' THEN (1 - DiscValue) \* BasePrice

WHEN 'QTY' THEN BasePrice

END AS DiscountedPrice,

CASE DiscType

WHEN 'CUR' THEN 0

WHEN 'PCT' THEN DiscValue \* 100

WHEN 'QTY' THEN 0

END AS DiscountPercent,

CASE DiscType

WHEN 'CUR' THEN 0

WHEN 'PCT' THEN 0

WHEN 'QTY' THEN DiscValue

END AS NumberToBuy,MinOrderAmt, FromDate, ToDate

FROM Menu M inner join MenuDisc "MD" on M.MenuID = "MD".MenuID

WHERE UPPER(Avail\_Ind) = 'A' AND M.MenuID NOT IN (SELECT \* FROM SetMeals) AND SYSDATE BETWEEN FromDate AND ToDate

ORDER BY MenuItemCat, MenuID;

clear columns

clear breaks

clear computes

ttitle off

btitle off





**4.2.3 Trigger 1: Update the Quantity For Ingredients in Branch Inventory when a Sales Order is created at restaurant**

**Purpose:** During Order Entry, a SaleOrder record and at least one SaleOrderItem record will be created. This trigger is fired after insert of each SaleOrderItem record. From the menu item in the SaleOrderItem record, the ingredients that made up the menu item sold are determined. For e,g, a menu item can be a cheese burger which is made up of: 1 bun, 1 slice of cheese and 1 meat patty. This information is obtained from the Recipe table. For each ingredient, its Quantity On Hand (QOH) in the Branch Inventory table is decremented. Note that the quantity of a given ingredient can be > 1 (e.g. a double cheeseburger has 2 slices of cheese). If so, the quantity to be decremented will have to be multiplied by this number. Moreover, if the order quantity for the menu item is > 1, then the QOH to be decremented will need to be further multiplied by the order quantity. The logic in this trigger does not allow the QOH to become negative because that would violate the database table constraint that was setup on the BranchInventory table.

**Trigger code:**

CREATE OR REPLACE TRIGGER reduce\_brninv\_QOH\_trig

AFTER INSERT ON SaleOrderItem

FOR EACH ROW

DECLARE

-- This cursor contains the ingredients that make up the item being bought

CURSOR IngredList\_cur IS

SELECT IngID, QtyPerMenu

FROM Recipe

WHERE Recipe.MenuID = :New.MenuID;

v\_IngID RECIPE.IngID%TYPE;

v\_QtyPerMenu RECIPE.QtyPerMenu%TYPE;

v\_OldQtyOnHand BRANCHINVENTORY.QtyOnHand%TYPE;

v\_NewQtyOnHand BRANCHINVENTORY.QtyOnHand%TYPE;

v\_BranchID SALEORDER.BranchID%TYPE;

BEGIN

-- Get BranchID from SaleOrder

SELECT SALEORDER.BranchID INTO v\_BranchID

FROM SALEORDER

WHERE SaleOrdID = :New.SaleOrdID;

OPEN IngredList\_cur;

FETCH IngredList\_cur INTO v\_IngID, v\_QtyPerMenu;

WHILE IngredList\_cur%FOUND

LOOP

-- First retrieve Current QOH

SELECT QtyOnHand INTO v\_OldQtyOnHand

FROM BranchInventory

WHERE BranchID = v\_BranchID AND BranchInventory.IngID = v\_IngID;

-- v\_QtyPerMenu is the quantity of the ingredient (e.g. 1 piece of cheese when buying a burger)

-- Using the burger example, :Qty is the number of burgers being bought

v\_NewQtyOnHand := v\_OldQtyOnHand - (v\_QtyPerMenu \* :New.Qty);

IF v\_NewQtyOnHand >= 0 THEN

dbms\_output.put\_line('New Quantity on Hand is '||v\_NewQtyOnHand||' for '||v\_IngID);

UPDATE BranchInventory

SET QtyOnHand = v\_NewQtyOnHand

WHERE BranchInventory.BranchID = v\_BranchID AND BranchInventory.IngID = v\_IngID;

END IF;

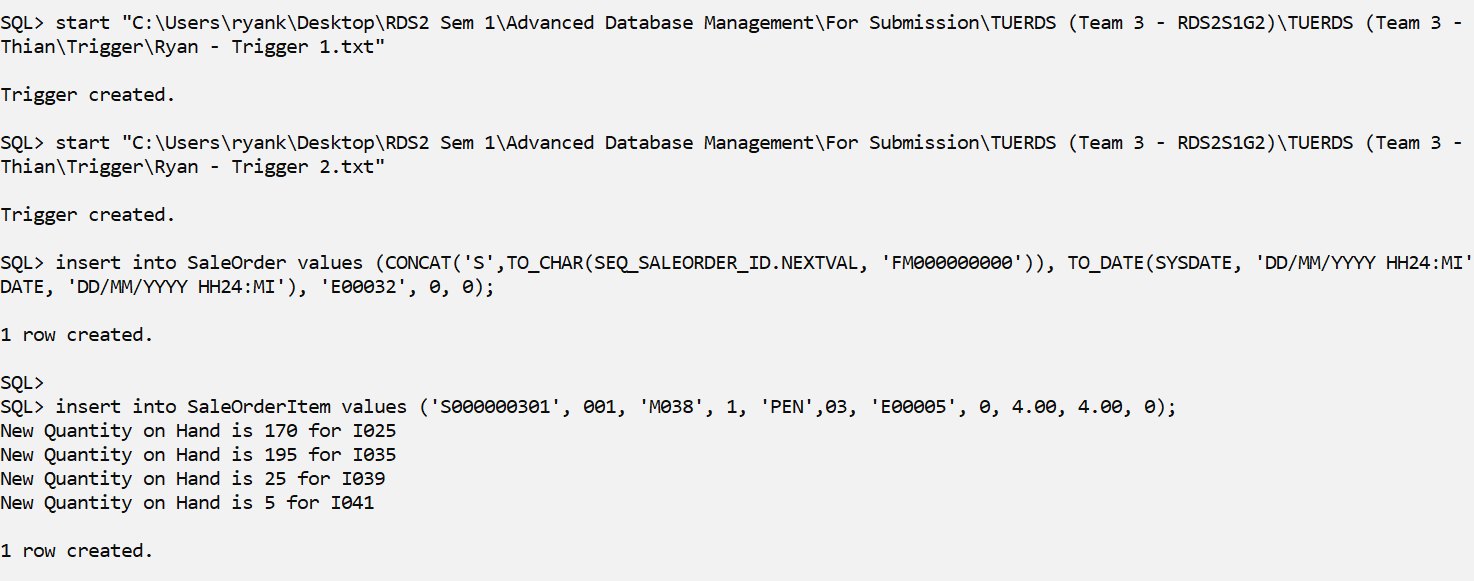
FETCH IngredList\_cur INTO v\_IngID, v\_QtyPerMenu;

END LOOP;

END;

/

**Sample Output:**

****

**\*\*Please take note that the message displayed is due to Trigger 1 after a sales order item record is inserted**

**4.2.4 Trigger 2: Create a PurchaseOrder for ingredient when its QuantityOnHand is less than or equal to its MinQty**

**Purpose:** This trigger is fired after a BranchInventory record is updated. If the Quantity On Hand of the item (ingredient) falls below or is equal to the Minimum Inventory Level (MinQty), the logic will first check if there is any pending purchase order for the ingredient whose QOH is being updated. If yes, the trigger will not create a new Purchase Order. This is to avoid ordering too much of the ingredient i.e. oversupply. Only if there is no pending purchase order for the ingredient, then a new Purchase Order is created. Since this Purchase Order is auto-created by the system and not by a human, certain field values are defaulted. The SupplierID is defaulted to a ‘dummy’ supplier (SP011) that we have setup. Later on, when the Purchase Order is picked up by a human purchaser, the human is expected to assign a different, real supplier to the Purchase Order. The Purchase Order Date is defaulted to the current system date. The unit Price and total Price are both set to 0 since the price is unknown until a supplier is contacted. Finally, the Quantity Ordered is set to 1 if there is no Minimum Inventory Level (MinQty = 0) is setup for the ingredient. Obviously, a human purchaser will have to change this order quantity from 1 to a bigger number. If a Minimum Inventory Level has been setup (i.e. MinQty > 0), the Quantity Ordered is equal to Minimum Inventory Level (MinQty) minus Quantity On Hand plus Minimum Inventory Level

(MinQty). The rationale for this formula is as follows. First, compute how much the Quantity On Hand is short of the Minimum Inventory Level (MinQty). Then add Minimum Inventory Level (MinQty) to this difference. So, in effect, we are trying to make the Quantity On Hand twice the Minimum Inventory Level (MinQty). This quantity should sustain additional sale orders for the ingredient for a while until QOH drops to MinQty or below. In any case, a human user can override this system-computed order quantity if he/she thinks it is too high or too low. The new Purchase Order will have the status ‘PA’ (Pending Assignment to supplier). This means the nextstep is for a human purchaser to review and update the Purchase Order with a real supplier and set the Order Quantity and Price to more appropriate values.

**Trigger code:**

CREATE OR REPLACE TRIGGER auto\_create\_PO\_trig

AFTER UPDATE ON BranchInventory

FOR EACH ROW

DECLARE

v\_count NUMBER := 0;

v\_ingredient\_qty NUMBER := 0;

BEGIN

-- If Quantity On Hand falls below or become equal to Min Quantity, create a Purchase Order

IF :New.QtyOnHand <= :New.MinQty THEN

-- Check if there already exists Pending purchase orders for this item and branch. If so, do not want to create another PO else we have an over-supply.

SELECT COUNT(\*) INTO v\_count FROM purchaseorder WHERE purchaseorder.BranchID = :New.BranchID AND purchaseorder.IngID = :New.IngID AND

purchaseorder.Status IN ('PA', 'PD');

IF v\_count = 0 THEN

IF :New.MinQty = 0 THEN

v\_ingredient\_qty := 1;

ELSE

v\_ingredient\_qty := :New.MinQty - :New.QtyOnHand + :New.MinQty;

END IF;

INSERT INTO PurchaseOrder VALUES ('SP011', :New.IngID, :New.BranchID, SYSDATE, NULL, v\_ingredient\_qty, 0, 0, 'PA');

END IF;

END IF;

END;

/

**Sample Output:**

There is no sample output here because this trigger is mainly used to perform validations and insert records into purchase order when certain requirements are fulfilled.

**4.2.5 Procedure 1: Allows a human user to update missing info (price, qty, supplier) in the Purchase Order created by my trigger. After that, the Purchase Order can be dispatched to supplier.**

**Purpose:**

For those Purchase Orders which were created by the Trigger, a human user needs to edit

some of the fields. When the trigger created a Purchase Order, the Supplier ID was defaulted to “SP011” which is a dummy supplier setup in the system. Therefore, the human needs to change the Supplier ID to a real supplier. Once the supplier has been selected, the human user needs to contact the supplier to get the Price of the item being ordered. Apart from this, the human user may also need to edit the Order Quantity because the value set by the trigger may not be appropriate. A purchase order is retrieved by entering the key fields which are: Branch ID, Supplier ID, Purchase Date and Ingredient ID. These fields will be prompted. After the fields’ values have been entered, if the purchase order exists, the purchase order’s current fields’ values will be displayed. The user will first be prompted to enter the new Supplier ID. Next, the user will be prompted for the Order Quantity and lastly, the Order Price. After all these values have been entered, the stored procedure will be called to update the Purchase Order in the database. The purchase order Status will be set to ‘PD’ (Pending Delivery).

**Procedure code:**

alter session set nls\_date\_format = 'DD/MM/YYYY';

CREATE OR REPLACE PROCEDURE update\_po (

key\_branch\_id\_parm IN PurchaseOrder.Branchid%TYPE,

key\_supplier\_id\_parm IN PurchaseOrder.supplierid%TYPE,

key\_purch\_date\_parm IN VARCHAR2,

key\_ingredient\_id\_parm IN PurchaseOrder.ingid%TYPE,

chg\_supplier\_id\_parm IN PurchaseOrder.supplierid%TYPE,

chg\_order\_qty\_parm IN NUMBER,

chg\_order\_price\_parm IN NUMBER

) IS

BEGIN

UPDATE purchaseorder SET PurchaseOrder.Status = 'PD', PurchaseOrder.SupplierID = chg\_supplier\_id\_parm,PurchaseOrder.IngredientQty = TO\_NUMBER(chg\_order\_qty\_parm), PurchaseOrder.BasePrice = TO\_NUMBER(chg\_order\_price\_parm), PurchaseOrder.PSubtotal = TO\_NUMBER(chg\_order\_price\_parm) \* TO\_NUMBER(chg\_order\_qty\_parm)

WHERE PurchaseOrder.BranchID = key\_branch\_id\_parm AND

PurchaseOrder.SupplierID = key\_supplier\_id\_parm AND

TO\_DATE(PurchaseOrder.PurchaseDate, 'DD/MM/YYYY') = TO\_DATE(key\_purch\_date\_parm,'DD/MM/YYYY') AND

PurchaseOrder.ingID = key\_ingredient\_id\_parm;

END;

/

--The User Interface to use this stored procedure

PROMPT

PROMPT =====================================================================

PROMPT Update Supplier, Qty and Price in Purchase Order

PROMPT =====================================================================

-- Prompt user to enter Key fields in order to retrieve Purchase Order

PROMPT Enter the Key Fields which identify the Purchase Order to be updated.

PROMPT

ACCEPT branch\_id PROMPT 'Branch ID > '

ACCEPT supplier\_id PROMPT 'Supplier ID > '

ACCEPT purch\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Purchase Date (DD/MM/YYYY) > '

ACCEPT ingredient\_id PROMPT 'Ingredient ID > '

SELECT \*

FROM PurchaseOrder

WHERE PurchaseOrder.BranchID = UPPER('&branch\_id') AND

PurchaseOrder.SupplierID = UPPER('&supplier\_id') AND

TO\_DATE(PurchaseOrder.PurchaseDate, 'DD/MM/YYYY') = TO\_DATE('&purch\_date','DD/MM/YYYY') AND

PurchaseOrder.IngID = UPPER('&ingredient\_id');

PROMPT Please enter additional details

ACCEPT new\_supplier\_id PROMPT 'New Supplier ID > '

ACCEPT order\_qty NUMBER FORMAT 9999999.99 PROMPT 'Order Quantity > '

ACCEPT order\_price NUMBER FORMAT 999.99 PROMPT 'Order Price (RM) > '

-- Call Stored Procedure to update Purchase Order

EXEC update\_po('&branch\_id', '&supplier\_id', '&purch\_date', '&ingredient\_id', '&new\_supplier\_id', &order\_qty, &order\_price);

/

**Sample Output:**



**4.2.6 Procedure 2:**

**This Stored Procedure processes a delivered purchase order. It increments the Quantity On Hand for the Ingredient that has arrived at the branch. The Status of the Purchase Order will be changed from 'PD' (Pending Delivery) to 'AR' (Arrived)**

**Purpose:**

When the goods for a Purchase Order have arrived, the purchase order Status needs to be

updated and the inventory incremented. The user will be prompted for the key fields of the

purchase order which are: Branch ID, Supplier ID, Purchase Date and Ingredient ID. After

retrieving the purchase order, the stored procedure will proceed to lookup and increment

the Quantity On Hand for the Branch Inventory record. After that, it will update the purchase

order status from ‘PD’ (Pending Delivery) to ‘AR’ (Arrived).

**Procedure code:**

CREATE OR REPLACE PROCEDURE process\_po\_delivery\_sp

(

branch\_id\_parm IN PurchaseOrder.Branchid%TYPE,

supplier\_id\_parm IN PurchaseOrder.supplierid%TYPE,

purch\_date\_parm IN varchar2,

ingredient\_id\_parm IN PurchaseOrder.ingid%TYPE

) AS

v\_status PURCHASEORDER.Status%TYPE;

v\_ingredientqty PURCHASEORDER.IngredientQty%TYPE;

v\_OldQtyOnHand BRANCHINVENTORY.QtyOnHand%TYPE;

v\_NewQtyOnHand BRANCHINVENTORY.QtyOnHand%TYPE;

BEGIN

SELECT Status, IngredientQty INTO v\_status, v\_ingredientqty

FROM PurchaseOrder

WHERE PurchaseOrder.BranchID = branch\_id\_parm AND

PurchaseOrder.SupplierID = supplier\_id\_parm AND

TO\_DATE(PurchaseOrder.PurchaseDate, 'DD/MM/YYYY') = TO\_DATE(purch\_date\_parm, 'DD/MM/YYYY') AND

PurchaseOrder.ingid = ingredient\_id\_parm;

IF v\_status = 'PD' THEN

SELECT QtyOnHand INTO v\_OldQtyOnHand

FROM BranchInventory

WHERE BranchID = branch\_id\_parm AND BranchInventory.IngID = ingredient\_id\_parm;

v\_NewQtyOnHand := v\_OldQtyOnHand + v\_ingredientqty;

UPDATE BranchInventory

SET QtyOnHand = v\_NewQtyOnHand

WHERE BranchID = branch\_id\_parm AND BranchInventory.IngID = ingredient\_id\_parm;

UPDATE PurchaseOrder

SET PurchaseOrder.Status = 'AR'

WHERE PurchaseOrder.BranchID = branch\_id\_parm AND

PurchaseOrder.SupplierID = supplier\_id\_parm AND

TO\_DATE(PurchaseOrder.PurchaseDate, 'DD/MM/YYYY') = TO\_DATE(purch\_date\_parm, 'DD/MM/YYYY') AND

PurchaseOrder.ingid = ingredient\_id\_parm;

dbms\_output.put\_line('Successfully processed purchase order delivery!');

dbms\_output.put\_line('Purchase order status is now AR (Arrived)');

ELSIF v\_status = 'AR' THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Error: Purchase Order Status is already AR (ARRIVED)');

ELSIF v\_status = 'PA' THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error: Purchase Order Status is only PA (Pending Assignment to Supplier)');

ELSE

RAISE\_APPLICATION\_ERROR(-20003, 'Error: Invalid Purchase Order');

END IF;

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No record found! You may have entered wrong value(s)!');

END;

/

--THE User Interface to use the stored proc

PROMPT

PROMPT =========================================================================================================

PROMPT Process Purchase Order Delivery

PROMPT =========================================================================================================

PROMPT To process a purchase order delivery, please enter the following details to retrieve the purchase order:

ACCEPT branch\_id PROMPT 'Branch ID > '

ACCEPT supplier\_id PROMPT 'Supplier ID > '

ACCEPT purch\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Purchase Date (DD/MM/YYYY) > '

ACCEPT ingredient\_id PROMPT 'Ingredient ID > '

EXEC process\_po\_delivery\_sp('&branch\_id', '&supplier\_id', '&purch\_date', '&ingredient\_id');

**Sample Output:**

**Scenario 1:Processing a delivered purchase order that has not been processed yet**



**Scenario 2: Attempting to process a delivered purchase order that has already been processed in Scenario 1 will cause an error**



**Scenario 3:Cannot process a purchase order that has not even been assigned to a valid supplier**



**Scenario 4:An error will occur if the purchase order does not exist**



**4.2.5 Report 1: On demand & Summary report listing Branches with Below Average Sales Revenues for the specified period of time. The Below Average Revenues are sorted from lowest to highest**

**Purpose:** This is a management report which lists the Branches which produced Below Average Sales for a selected period. It allows management to identify the poor performing branches. The overall average sales amount is displayed at the top of the report. By having this info, the management can get the managers of these branches to increase sales for e.g. by having promotions, special events or renovating to provide a nicer eating environment. Alternatively, if a branch consistently showed poor sales, the management may decide to close the branch. Technical Note: This report could have been produced by using purely SQL i.e. a subquery which aggregates sales by branch and then apply the AVG function. An outer select statement then aggregates sales by branch again and then compare each branch’s sales against the overall average sales. On the other hand, the cursor approach only needs to perform the aggregation of sales by branches only once. Let’s say there are 10 branches so the cursor contains only 10 rows. To compute the overall average sales, only 10 rows need to be read. Then in order to select the below average performing branches, only 10 rows need to be read. Compared to the pure SQL approach which needs to perform grouping on potentially huge numbers of sales order records 2 times, the cursor approach may be much more efficient.

**Report code:**

alter session set nls\_date\_format = 'DD/MM/YYYY';

set serveroutput on;

CREATE OR REPLACE PROCEDURE poor\_perform\_branches\_rpt (fromdate\_parm IN varchar2, thrudate\_parm IN varchar2) AS

v\_BranchID BRANCH.BranchID%TYPE;

v\_BranchName BRANCH.BranchName%TYPE;

v\_BranchTotalRev NUMBER(13,2);

v\_AvgBrnSlsRev NUMBER(13,2);

v\_NoOfBranches NUMBER(13,2);

v\_OverallSlsRev NUMBER(13,2);

v\_fromdate DATE;

v\_thrudate DATE;

cursor BrnSlsRev\_cursor is

select B.BranchID, BranchName, SUM(GrandTotal) as BranchTotalRev

from SALEORDER S inner join BRANCH B on S.BranchID = B.BranchID

where UPPER(SaleOrdStat) in ('COM','COL') and SaleOrdDate between TO\_DATE(fromdate\_parm, 'DD/MM/YYYY') AND TO\_DATE(thrudate\_parm, 'DD/MM/YYYY')

group by B.BranchID, BranchName

order by BranchTotalRev;

BEGIN

v\_BranchTotalRev := 0;

v\_AvgBrnSlsRev := 0;

v\_NoOfBranches := 0;

v\_OverallSlsRev := 0;

v\_fromdate := TO\_DATE(fromdate\_parm,'DD/MM/YYYY');

v\_thrudate := TO\_DATE(thrudate\_parm,'DD/MM/YYYY');

IF v\_thrudate >= v\_fromdate THEN

-- Loop thru cursor to calculate Average Branch Sales Revenue

OPEN BrnSlsRev\_cursor;

FETCH BrnSlsRev\_cursor INTO v\_BranchID, v\_BranchName, v\_BranchTotalRev;

WHILE BrnSlsRev\_cursor%FOUND

LOOP

v\_NoOfBranches := v\_NoOfBranches + 1;

v\_OverallSlsRev := v\_OverallSlsRev + v\_BranchTotalRev;

FETCH BrnSlsRev\_cursor INTO v\_BranchID, v\_BranchName, v\_BranchTotalRev;

END LOOP;

CLOSE BrnSlsRev\_cursor;

IF v\_NoOfBranches > 0 THEN

dbms\_output.put\_line(chr(10));

dbms\_output.put\_line(LPAD('=', 60, '='));

dbms\_output.put\_line('Branches with Sales Revenues Below Average (Summary)');

dbms\_output.put\_line(LPAD('=', 60, '='));

dbms\_output.put\_line('Time Period: '||fromdate\_parm||' to '||thrudate\_parm );

v\_AvgBrnSlsRev := v\_OverallSlsRev/v\_NoOfBranches;

dbms\_output.put\_line('Average Branch Sales Revenue: RM '|| v\_AvgBrnSlsRev);

dbms\_output.put\_line(chr(10));

dbms\_output.put\_line(RPAD('Branch No',11) || RPAD('Branch Name', 20) || LPAD('Sales Revenue', 15));

dbms\_output.put\_line(LPAD('=', 50, '='));

- Re-open cursor to select only branches whose sales revenues are below average

OPEN BrnSlsRev\_cursor;

FETCH BrnSlsRev\_cursor INTO v\_BranchID, v\_BranchName, v\_BranchTotalRev;

WHILE BrnSlsRev\_cursor%FOUND

LOOP

IF v\_BranchTotalRev < v\_AvgBrnSlsRev THEN dbms\_output.put\_line(RPAD(v\_BranchID, 11)|| RPAD(v\_BranchName, 20)||' '|| 'RM ' || ROUND(v\_BranchTotalRev, 2));

dbms\_output.put\_line(chr(10));

END IF;

FETCH BrnSlsRev\_cursor INTO v\_BranchID, v\_BranchName, v\_BranchTotalRev;

END LOOP;

CLOSE BrnSlsRev\_cursor;

dbms\_output.put\_line('===================== End of Report ========================');

END IF;

ELSE

RAISE\_APPLICATION\_ERROR(-20000, 'Error: FROM Date cannot be later than TO Date');

END IF;

END;

/

--To get input to generate report

/\* Prompt user for FROM and TO dates and pass to stored procedure \*/

PROMPT Generate a Listing of Branches with Below Average Sales Revenues for a specified period

PROMPT -----------------------------------------------------------------------------------------

ACCEPT from\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter FROM Date: '

ACCEPT thru\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter TO Date : '

-- Call Stored Procedure

EXEC poor\_perform\_branches\_rpt('&&from\_date', '&&thru\_date');

**Sample Output:**



**4.2.6 Report 2: Detailed report showing Suppliers’ Average Turnaround Time (in Days) for Deliveries, sorted from the Shortest Average Turnaround Time to the Longest Average Turnaround Time**

**Purpose:** This report lists all suppliers by ascending average turnaround time (fastest to slowest). Turnaround time in days is computed by subtracting Purchase Order Date from Delivery Date. Besides the average turnaround time, the report also displays which ingredients or raw materials we order from each supplier. With this info, management can determine which suppliers are more responsive than others and therefore decisions can be made as to which suppliers to give the company’s business to. Technical Note: This report uses nested cursors, one at the supplier level and another for items supplied by the supplier. This type of report format, with the average turnaround time displayed at the supplier header (not footer), is awkward or maybe not even possible to construct using pure SQL. Putting the average turnaround time at the footer of the supplier block would be easy to do using pure SQL and SQL Plus. However, doing that introduces an extra line (footer) and the user would need to scan down to look for the footer in order to find the average turnaround time. Planing the average turnaround time at the supplier header (first line of supplier block) can be easily accomplished using nested cursors. After reading a row from the suppliers cursor with average turnaround time, the items supplied by the supplier can be retrieved from the second cursor.

**Report code:**

set serveroutput on;

CREATE OR REPLACE PROCEDURE suppliers\_turnaround\_rpt AS

v\_SupplierID SUPPLIER.SupplierID%TYPE;

v\_SupplierName SUPPLIER.SupplierName%TYPE;

v\_AvgNoDays NUMBER(3);

v\_IngID INGREDIENT.IngID%TYPE;

v\_IngName INGREDIENT.IngName%TYPE;

v\_counter number(3);

cursor supp\_turnard\_cursor is

select S.SupplierID, SupplierName, ROUND(AVG(DeliveredDate - PurchaseDate),0) as AvgNoDays

from PURCHASEORDER P inner join SUPPLIER S on P.SupplierID = S.SupplierID

where Status = 'AR'

group by S.SupplierID, SupplierName

order by AvgNoDays;

cursor supp\_ingredient\_cursor is

select I.IngID, IngName

from PURCHASEORDER P inner join INGREDIENT I on P.IngID = I.IngID

where P.SupplierID = v\_SupplierID

group by I.IngID, IngName;

BEGIN

OPEN supp\_turnard\_cursor;

FETCH supp\_turnard\_cursor INTO v\_SupplierID, v\_SupplierName, v\_AvgNoDays;

v\_counter := 1;

dbms\_output.put\_line(LPAD('=', 55, '='));

dbms\_output.put\_line('Suppliers Average Turnaround Time (Days) for Deliveries');

dbms\_output.put\_line(LPAD('=', 55, '='));

dbms\_output.put\_line('From Fastest to Slowest');

dbms\_output.put\_line(chr(10));

WHILE supp\_turnard\_cursor%FOUND

LOOP

dbms\_output.put\_line(v\_counter || ') ' || 'Supplier '|| ': ' ||v\_SupplierName || ' (' || v\_SupplierID || ')');

dbms\_output.put\_line('Average Turnaround Time (Days) : '||v\_AvgNoDays);

dbms\_output.put\_line(LPAD('\*', 40, '\*'));

OPEN supp\_ingredient\_cursor;

FETCH supp\_ingredient\_cursor INTO v\_IngID, v\_IngName;

WHILE supp\_ingredient\_cursor%FOUND

LOOP

dbms\_output.put\_line('> Ingredient: '||v\_IngName || ' (' || v\_IngID || ')');

FETCH supp\_ingredient\_cursor INTO v\_IngID, v\_IngName;

END LOOP;

CLOSE supp\_ingredient\_cursor;

v\_counter := v\_counter + 1;

dbms\_output.put\_line(LPAD('\*', 40, '\*'));

dbms\_output.put\_line(chr(10));

FETCH supp\_turnard\_cursor INTO v\_SupplierID, v\_SupplierName, v\_AvgNoDays;

END LOOP;

CLOSE supp\_turnard\_cursor;

END;

/

--To generate report

exec suppliers\_turnaround\_rpt

**Sample Output:**











**4.3 (Sim Hong Lii)**

**4.3.1 Query 1: Ranking of primary delivery companies from each branch**

**Purpose: The query to retrieve the top 3 primary delivery companies which are kfc delivery, grad food and food panda from each branch allows the restaurants to optimize its delivery network. By identifying the most frequently utilized delivery companies for each branch, the company gains insights into the effectiveness of its delivery partners**

**SQL statement:**

BREAK ON BranchID ON BranchName NODUP SKIP 1

COLUMN BranchID FORMAT A15 HEADING "Branch ID";

COLUMN BranchName FORMAT A16 HEADING "Branch Name";

COLUMN DeliveryCompanyID FORMAT A20 HEADING "Delivery Company ID";

COLUMN DeliveryCompanyName FORMAT A21 HEADING "Delivery Company Name";

COLUMN COUNT(V.SaleOrdID) FORMAT A15 HEADING "Number Of Orders";

TTITLE CENTER 'ALL TIME POPULAR Primary Delivery Company In each Branch'-

RIGHT 'Page:' FORMAT 999 SQL.PNO SKIP 2

SELECT

B.BranchID, B.BranchName,

V.DeliveryCompanyID,

V.DeliveryCompanyName,

COUNT(V.SaleOrdID) AS NumberOfOrders

FROM

Branch B, SaleOrder SO , Popular\_Delivery\_company\_VIEW V

where

B.BranchID = SO.BranchID

and

SO.SaleOrdID = V.SaleOrdID

GROUP BY

B.BranchID, B.BranchName,

V.DeliveryCompanyID,

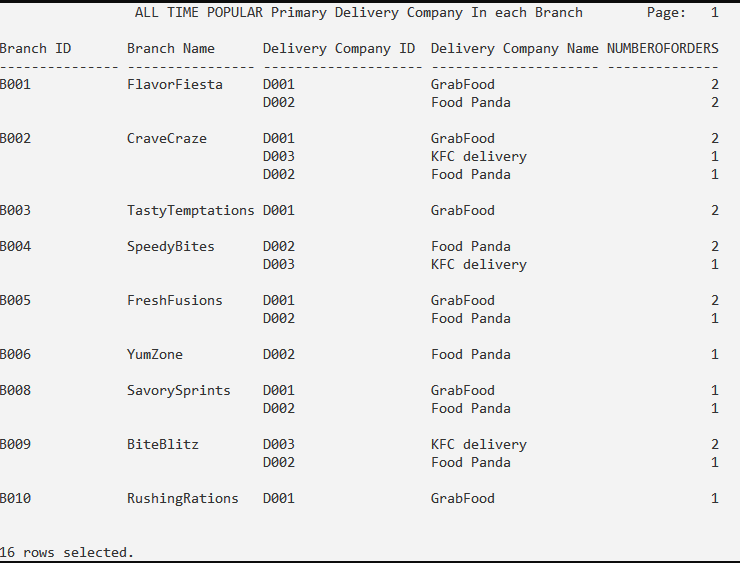
V.DeliveryCompanyName

ORDER BY

B.BranchID,

NumberOfOrders DESC;

**Sample Output:**



**4.3.2 Query 2: Ranking least purchase ingredient of fresh food in purchases Catalog**

**Purpose: By analyzing the purchasing patterns of these essential ingredients, the company can optimize its supply chain and procurement processes. It can monitoring the usage of less frequently purchased items (least purchase). This information can lead to cost savings, reduced waste, and improved menu planning, ultimately helping the company efficiently manage its resources and meet customer demand for fresh food items. Additionally, it can help identify potential issues with suppliers or product availability, allowing for timely adjustments in procurement strategies.**

**SQL statement:**

SET PAGESIZE 30

SET LINESIZE 210

COLUMN SupplierName FORMAT A22 HEADING "Supplier Name";

COLUMN IngID FORMAT A13 HEADING "Ingredient ID";

COLUMN IngName FORMAT A26 HEADING "Ingredient Name";

COLUMN IngType FORMAT A20 HEADING "Ingredient Type";

COLUMN SUM(PO.IngredientQTY) FORMAT A15 HEADING "Number of Purchased";

COLUMN ContactNo FORMAT A11 HEADING "Contact No";

COLUMN DUMMY NOPRINT

BREAK ON DUMMY SKIP 1

BREAK ON SupplierName ON ContactNo NODUP SKIP 1

TTITLE Left 'Least Purchase Ingredient Of Fresh Food Supplied ' 'Page:' FORMAT 999 SQL.PNO SKIP 2

SELECT \*

FROM (

SELECT S.SupplierName, V.IngID, V.IngName, V.IngType, SUM(PO.IngredientQTY) AS Purchased, S.ContactNo

FROM PurchaseOrder PO, fresh\_ingredient\_VIEW V, Supplier S

WHERE PO.IngID = V.IngID and PO.SupplierID = S.SupplierID

GROUP BY V.IngID, V.IngName, V.IngType, S.SupplierName, S.ContactNo

ORDER BY SUM(PO.IngredientQTY), V.IngID

)

WHERE ROWNUM <= 10;

**Sample Output:**



**4.3.3 Procedure 1: Add Purchase**

**Purpose:The purpose of this procedure is to purchase new ingredient in the Purchasing Catalog. In order to add a new purchase, user must ensure that the purchase being added is valid by keying in the branch id ingid, purchase date and delivery date along with the purchase details of the ingredient. Based on Query1 we know which ingredient is lowest so I plan to add it to prevent emergency.**

**Procedure code:**

drop PROCEDURE pro\_add\_PurchaseOrder;

CREATE OR REPLACE PROCEDURE pro\_add\_PurchaseOrder(

IN\_SupplierID IN PurchaseOrder.SupplierID%TYPE,

IN\_IngID IN PurchaseOrder.IngID%TYPE,

IN\_BranchID IN PurchaseOrder.BranchID%TYPE,

IN\_PurchaseDate IN PurchaseOrder.PurchaseDate%TYPE,

IN\_DELIVEREDDATE IN PurchaseOrder.DELIVEREDDATE%TYPE,

IN\_IngredientQty IN PurchaseOrder.IngredientQty%TYPE,

IN\_BasePrice IN PurchaseOrder.BasePrice%TYPE,

IN\_PSubtotal IN PurchaseOrder.PSubtotal%TYPE,

IN\_Status IN PurchaseOrder.Status%TYPE

) is

v\_SupplierID PurchaseOrder.PurchaseDate%TYPE;

v\_IngID PurchaseOrder.IngID%TYPE;

v\_PurchaseDate PurchaseOrder.PurchaseDate%TYPE;

INSERT\_WRONG\_SupplierID EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_SupplierID, -20001);

INSERT\_WRONG\_PurchaseDate EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_PurchaseDate, -20002);

INSERT\_WRONG\_IngreidientID EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_IngreidientID, -20003);

INSERT\_WRONG\_BranchID EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_BranchID, -20004);

cursor Purchase\_cursor is

select SupplierID, IngID, PurchaseDate

from PurchaseOrder

WHERE SupplierID = IN\_SupplierID and IngID = IN\_IngID and IN\_PurchaseDate = v\_PurchaseDate;

BEGIN

OPEN Purchase\_cursor;

FETCH Purchase\_cursor INTO v\_SupplierID, v\_IngID, v\_PurchaseDate;

IF IN\_SupplierID > 'SP011' or IN\_SupplierID < 'SP001' THEN

RAISE\_APPLICATION\_ERROR(-20001,'Supplier ID insert data wrongly.', true);

END IF;

IF IN\_PurchaseDate > IN\_DELIVEREDDATE THEN

RAISE\_APPLICATION\_ERROR(-20002,'Purchase Date insert data wrongly it, it must not after Delivery Date.', true);

end if;

IF IN\_IngID < 'I001' or IN\_IngID > 'I041' THEN

RAISE\_APPLICATION\_ERROR(-20003,'Ingredient ID insert data wrongly.', true);

END IF;

IF IN\_BranchID < 'B001' or IN\_BranchID > 'B010' THEN

RAISE\_APPLICATION\_ERROR(-20004,'Branch ID insert data wrongly.', true);

END IF;

INSERT INTO PurchaseOrder VALUES (IN\_SupplierID, IN\_IngID ,IN\_BranchID, IN\_PurchaseDate, IN\_DELIVEREDDATE, IN\_IngredientQty,IN\_BasePrice ,IN\_PSubtotal, IN\_Status);

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE('Purchase successfully add in.');

DBMS\_OUTPUT.PUT\_LINE(RPAD('Supplier ID', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_SupplierID), 5));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Ingredient ID', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_IngID), 4));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Branch ID', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_BranchID), 4));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Purchase Date', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_PurchaseDate), 10, '0'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Delivery Date', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_DELIVEREDDATE), 10, '0'));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Ingredient Qty', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_IngredientQty), 3));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Base Price', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_BasePrice), 3));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Subtotal', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_PSubtotal), 3));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Purchase Status', 20, ' ') || ': ' || LPAD(TO\_CHAR(IN\_Status), 2));

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

EXCEPTION

WHEN INSERT\_WRONG\_SupplierID THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE('Failed to print report for ' || IN\_SupplierID || '.');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

WHEN INSERT\_WRONG\_PurchaseDate THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE('Failed to print report for ' || IN\_PurchaseDate || '.');

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

WHEN INSERT\_WRONG\_IngreidientID THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE('Failed to print report for ' || IN\_IngID || '.');

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

WHEN INSERT\_WRONG\_BranchID THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE('Failed to print report for ' || IN\_BranchID || '.');

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

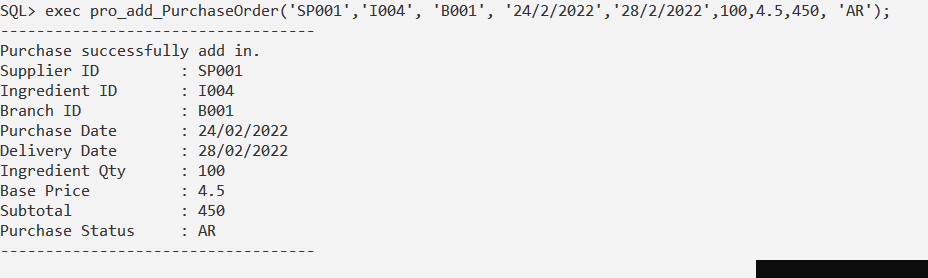
END;

/

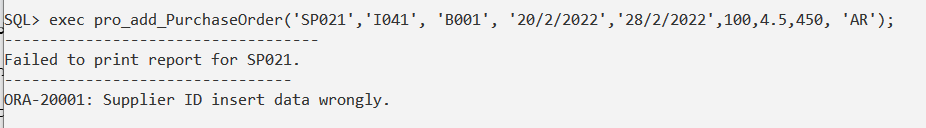
exec pro\_add\_PurchaseOrder('SP001','I004', 'B001', '24/2/2022','28/2/2022',100,4.5,450, 'AR');

--exec pro\_add\_PurchaseOrder('SP021','I041', 'B001', '20/2/2022','28/2/2022',100,4.5,450, 'AR');

**Sample Output:**



EXCEPTION:



**4.3.4 procedure 2: Cancel delivery order**

**Purpose: Delete or cancel a delivery order, that will based on sales order status cancel delivery info, this provides a flexibility for those customers who misoder. The cancel only approve when status is 'pending', after approval it will insert the record in the cancel delivery table and delete the record in diliveryoderinfo table.**

**Procedure code:**

set serveroutput on

drop TABLE Delivery\_Cancel;

CREATE TABLE Delivery\_Cancel

( IN\_DsaleID CHAR(10),

V\_PayRefNo CHAR(9),

V\_MenuID CHAR(4),

V\_Qty number(3),

V\_SalesTax NUMBER (5,3),

V\_PaidAmt NUMBER(8,2),

V\_RefundAmt NUMBER(8,2),

V\_PaymenSta char(1)

);

CREATE OR REPLACE PROCEDURE PRC\_RemoveDel(IN\_DsaleID IN CHAR) IS

V\_SaleOrdStat SaleOrder.SaleOrdStat%TYPE;

V\_PayRefNo Payment.PayRefNo%TYPE;

V\_MenuID SaleOrderItem.MenuID%TYPE;

V\_Qty SaleOrderItem.Qty%TYPE;

V\_SalesTax SaleOrder.SalesTax%TYPE;

V\_PaidAmt Payment.PaidAmt%TYPE;

V\_RefundAmt NUMBER(8,2);

V\_PaymenSta char(1);

INSERT\_WRONG\_SalesOderId EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_SalesOderId, -20006);

CURSOR Order\_CURSOR IS

SELECT

S.SaleOrdStat,

P.PayRefNo,

SI.MenuID,

SI.Qty,

S.SalesTax,

P.PaidAmt

FROM DeliveryOrderInfo D, SaleOrder S, SaleOrderItem SI, payment P

WHERE D.SaleOrdID = S.SaleOrdID AND

SI.SaleOrdID = S.SaleOrdID AND

P.SaleOrdID = S.SaleOrdID AND

D.SaleOrdID = IN\_DsaleID;

BEGIN

OPEN Order\_CURSOR;

LOOP

FETCH Order\_CURSOR INTO

V\_SaleOrdStat, V\_PayRefNo, V\_MenuID, V\_Qty, V\_SalesTax, V\_PaidAmt;

EXIT WHEN Order\_CURSOR%NOTFOUND;

END LOOP;

IF V\_SaleOrdStat = 'PEN' THEN

V\_RefundAmt := V\_PaidAmt - 5;

V\_PaymenSta := 'R';

--Write date to Delivery\_Cancel table

INSERT INTO Delivery\_Cancel VALUES(IN\_DsaleID, V\_PayRefNo, V\_MenuID, V\_Qty, V\_SalesTax, V\_PaidAmt, V\_RefundAmt,V\_PaymenSta);

DELETE FROM DeliveryOrderInfo

WHERE SaleOrdID = IN\_DsaleID;

DBMS\_OUTPUT.PUT\_LINE('DeliveryOrderInfo record with SaleOrdID ' || IN\_DsaleID || ' deleted successfully.');

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',60,'='));

DBMS\_OUTPUT.PUT\_LINE('Order status : '|| V\_SaleOrdStat);

DBMS\_OUTPUT.PUT\_LINE('Menu Id : '|| V\_MenuID);

DBMS\_OUTPUT.PUT\_LINE('Payment Refer No : '|| V\_PayRefNo);

DBMS\_OUTPUT.PUT\_LINE('Menu Quantity : '|| V\_Qty);

DBMS\_OUTPUT.PUT\_LINE('Sales Tax : '|| V\_SalesTax);

DBMS\_OUTPUT.PUT\_LINE('Paid Amount : '|| V\_PaidAmt);

DBMS\_OUTPUT.PUT\_LINE('Payment Status : '|| V\_PaymenSta);

DBMS\_OUTPUT.PUT\_LINE('Pending Refund amount on '||IN\_DsaleID|| ' is RM'||V\_PaidAmt);

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',60,'='));

else

RAISE\_APPLICATION\_ERROR(-20006,'You are not allowed deleted delivery. ', true);

END IF;

EXCEPTION

WHEN INSERT\_WRONG\_SalesOderId THEN

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 60, '-'));

DBMS\_OUTPUT.PUT\_LINE('You are not allowed deleted delivery ' || IN\_DsaleID || '.');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

exec PRC\_RemoveDel ('S004000261')

exec PRC\_RemoveDel ('S000000262')

exec PRC\_RemoveDel ('S000000115')

**Sample Output:**

**from**

| From the output in 4.3.1 Query 1, we can know that... | **CASE 1**: Top delivery company in BRandID **B009** is KFC delivery |
| --- | --- |
| select \*from Delivery\_Cancel; | Delivery order deleted insert in Delivery\_Cancel table |
|  | KFC delivery not the top delivery company in BRandID **B009** |
|  | **CASE 2**: SaleOrdID S004000261 **not exist so don't approve**. |

**4.3.5 Trigger 1: Check max of ingredient purchases**

**Purpose: The purpose of this trigger is to monitor the quantity of ingredient stored in the inventory as per the business rules of limiting the maximum inventory of each material in each branch is the sum of the material's purchase catalog in that branch plus two. If the purchase quantity of each raw material exceeds its raw material inventory limit, the purchase program will be refused to be added.**

**Trigger code:**

CREATE OR REPLACE TRIGGER TRG\_CHK\_MAX\_PurQTY

BEFORE INSERT ON PurchaseOrder

FOR EACH ROW

DECLARE

V\_TOTAL\_purQTY NUMBER;

G\_TOTAL\_purQTY NUMBER;

MAX\_QTY NUMBER;

BEGIN

SELECT SUM(IngredientQTY)

INTO V\_TOTAL\_purQTY

FROM PurchaseOrder

WHERE IngID = :NEW.IngID and BranchID = :NEW.BranchID;

MAX\_QTY := V\_TOTAL\_purQTY + 2;

G\_TOTAL\_purQTY := V\_TOTAL\_purQTY + :NEW.IngredientQTY;

IF SQL%FOUND THEN

IF V\_TOTAL\_purQTY + :NEW.IngredientQTY >= MAX\_QTY THEN

RAISE\_APPLICATION\_ERROR(-20900, 'Inventory is full for IngID: ' || :NEW.IngID || 'In Branch ' || :NEW.BranchID || ', Purchase Catalog Qty is ' || G\_TOTAL\_purQTY || ', larger than Max Inventory Qty ' || MAX\_QTY);

END IF;

ELSE

RAISE\_APPLICATION\_ERROR(-20951, 'PurchaseOrder INSERTION CURRENTLY UNAVAILABLE.');

END IF;

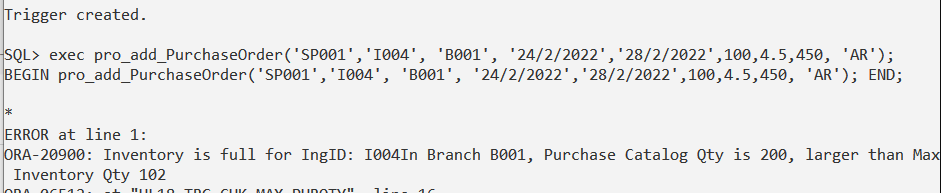
EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('INSERTION CURRENTLY UNAVAILABLE.');

END;

/



**4.3.6 Trigger 2: Update payment status of cancel delivery order**

**Purpose: The purpose of this trigger is to update the Payment table before the delivery order is deleted. This trigger will update the column payment status in Payment table from ‘P’ to ‘R’, ‘R’ means that this payment is a refund.**

**Trigger code:**

CREATE OR REPLACE TRIGGER TRG\_UPDATE\_cancel\_Payment

before INSERT

ON Delivery\_Cancel

FOR EACH ROW

DECLARE

V\_cancel\_PayRefNo Payment.PayRefNo%TYPE;

BEGIN

V\_cancel\_PayRefNo := :new.V\_PayRefNo;

IF :new.V\_PaymenSta = 'R' THEN

UPDATE Payment

SET PaymentStatus = 'R'

WHERE PayRefNo = V\_cancel\_PayRefNo;

END IF;

END;

/

**4.3.7 Report 1: Report on monthly total purchases of each branch in a Year(Summary)**

**Purpose: The purpose of this report is to generate a summary table each month on a specific year, and sum up each month how many purchase and list the specific ingredient already purchased. Through this report let management know some summary report in a specific year, to help in making decisions on the long term.**

**Report code:**

SET LINESIZE 500

SET PAGESIZE 125

CREATE OR REPLACE procedure prc\_Total\_Purchase\_each\_Month (IN\_year IN number) IS

RECORD\_FOUND EXCEPTION;

PRAGMA EXCEPTION\_INIT(RECORD\_FOUND , -20009);

Precord\_count NUMBER := 0;

SUM\_Poder NUMBER := 0;

CURSOR month\_cursor IS

SELECT DISTINCT EXTRACT(MONTH FROM PurchaseDate) as Month

FROM PurchaseOrder

ORDER BY Month ;

CURSOR supplier\_purchase\_cursor(M IN NUMBER) IS

SELECT

P.BranchID,

I.IngName,

P.PurchaseDate,

EXTRACT(MONTH FROM P.PurchaseDate) AS PurchaseMonth,

EXTRACT(YEAR FROM P.PurchaseDate) AS PurchaseYear,

P.DeliveredDate,

P.IngredientQty,

P.BasePrice,

P.PSubtotal,

S.SupplierName

FROM

Ingredient I,

Supplier S,

PurchaseOrder P

WHERE

P.SupplierID = S.SupplierID

AND P.IngID = I.IngID

AND EXTRACT(MONTH FROM P.PurchaseDate) = M

AND EXTRACT(YEAR FROM P.PurchaseDate) = IN\_year

ORDER BY

P.BranchID, P.PurchaseDate;

BEGIN

SELECT COUNT(PurchaseDate) INTO SUM\_Poder

FROM PurchaseOrder

WHERE EXTRACT (YEAR FROM PurchaseDate) = IN\_year;

IF SUM\_Poder = 0 THEN

RAISE\_APPLICATION\_ERROR(-20009, 'No record found.', true);

ELSE

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(chr(9) || chr(9) || chr(9) || chr(9)||chr(9)||'Summary of total purchase for each month year in '|| IN\_year);

DBMS\_OUTPUT.PUT\_LINE(chr(9) || chr(9) || chr(9) || chr(9)||chr(9)|| RPAD('\_',53,'\_'));

FOR mon\_purchase IN month\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Month', 10, ' ') || ': ' || RPAD(mon\_purchase.Month, 5, ' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 180, '='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Branch ID', 15,' ')|| RPAD('Ingredient Name', 30,' ') || RPAD('Purchase Date', 20,' ') || RPAD('Delivered Date', 20,' ') || RPAD('Ingredient Qty', 20,' ') || RPAD('Base Price', 15,' ') || RPAD('Subtotal', 20,' ') || RPAD('Supplier Name', 20,' '));

DBMS\_OUTPUT.PUT\_LINE(LPAD('-', 180, '-'));

FOR PurchaseOrder IN supplier\_purchase\_cursor(mon\_purchase.Month) LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(PurchaseOrder.BranchID, 15, ' ') || RPAD(PurchaseOrder.IngName, 30, ' ') || RPAD(TO\_CHAR(PurchaseOrder.PurchaseDate, 'DD/MM/YYYY'),20, ' ') || RPAD(TO\_CHAR(PurchaseOrder.DeliveredDate, 'DD/MM/YYYY'),20, ' ')||RPAD(PurchaseOrder.IngredientQty, 20, ' ')|| RPAD(PurchaseOrder.BasePrice, 15, ' ')||RPAD(PurchaseOrder.PSubtotal, 20, ' ') || RPAD(PurchaseOrder.SupplierName, 30, ' '));

Precord\_count := Precord\_count+1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Total PurchaseOrder in month ', 30, ' ') || RPAD(mon\_purchase.Month, 3, ' ')|| ': '|| RPAD(Precord\_count, 2, ' ') || ' Rows ');

DBMS\_OUTPUT.PUT\_LINE(RPAD('-', 35, '-'));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(LPAD('=', 180, '='));

END IF;

DBMS\_OUTPUT.PUT\_LINE(chr(10));

DBMS\_OUTPUT.PUT\_LINE( chr(9) ||chr(9)||chr(9)||chr(9)||chr(9)||'Total purchase for each month in year '|| IN\_year || ': ' || RPAD(SUM\_Poder, 5, ' '));

EXCEPTION

WHEN RECORD\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Failed to print report for ' || IN\_year || '.');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

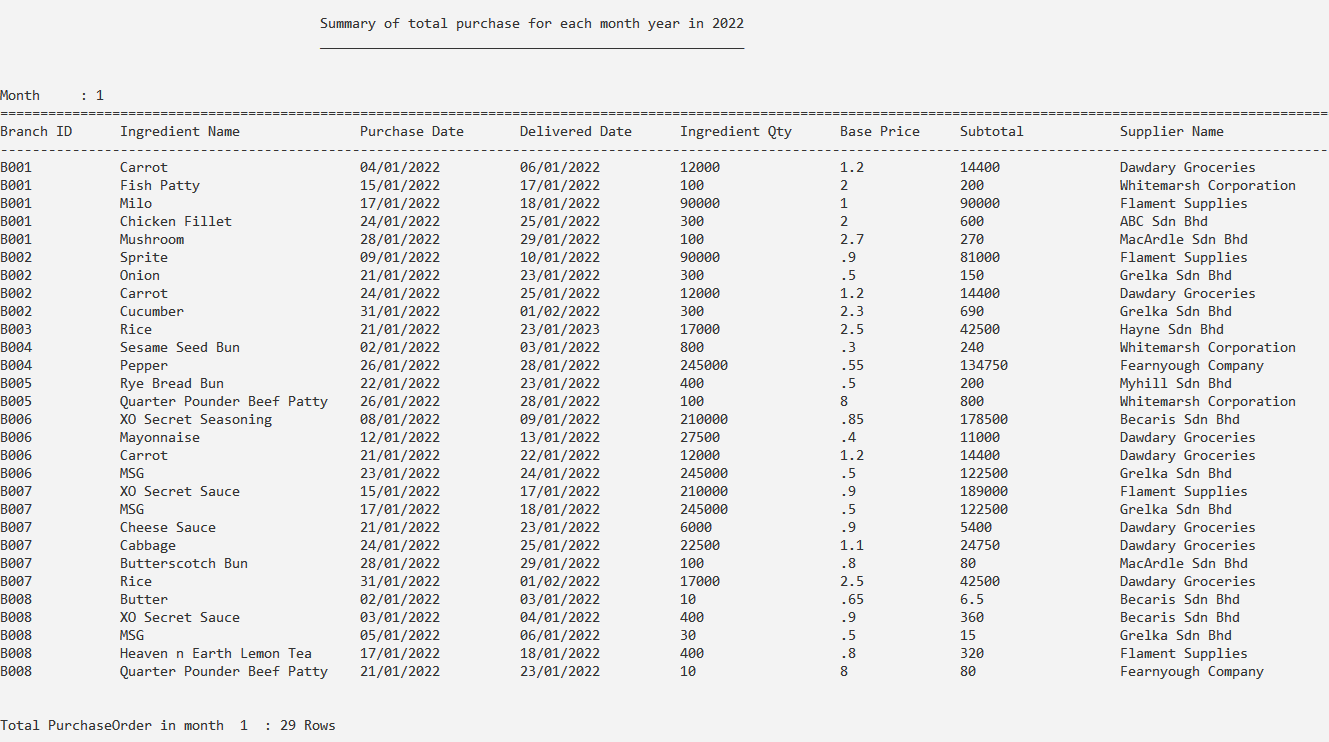
DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

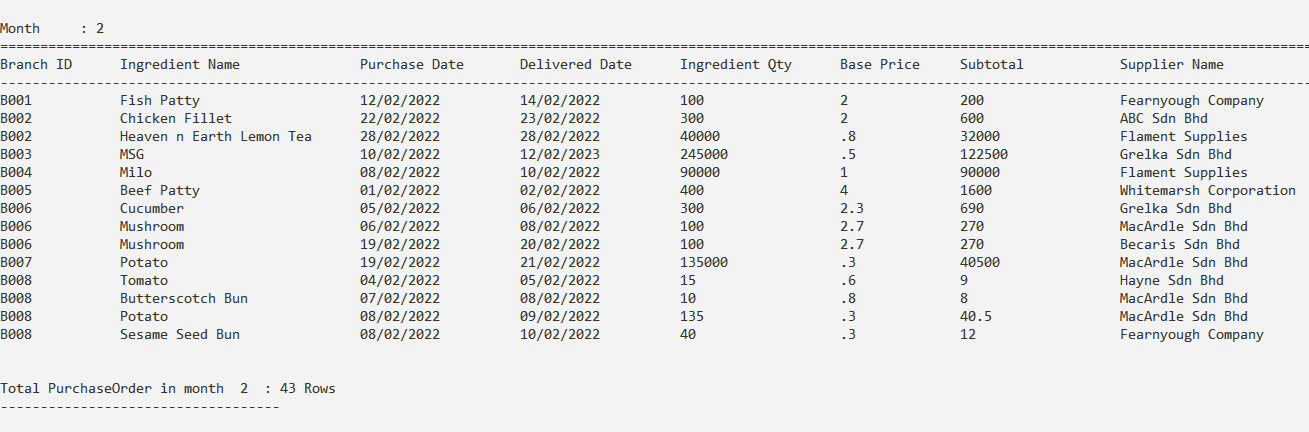
END;

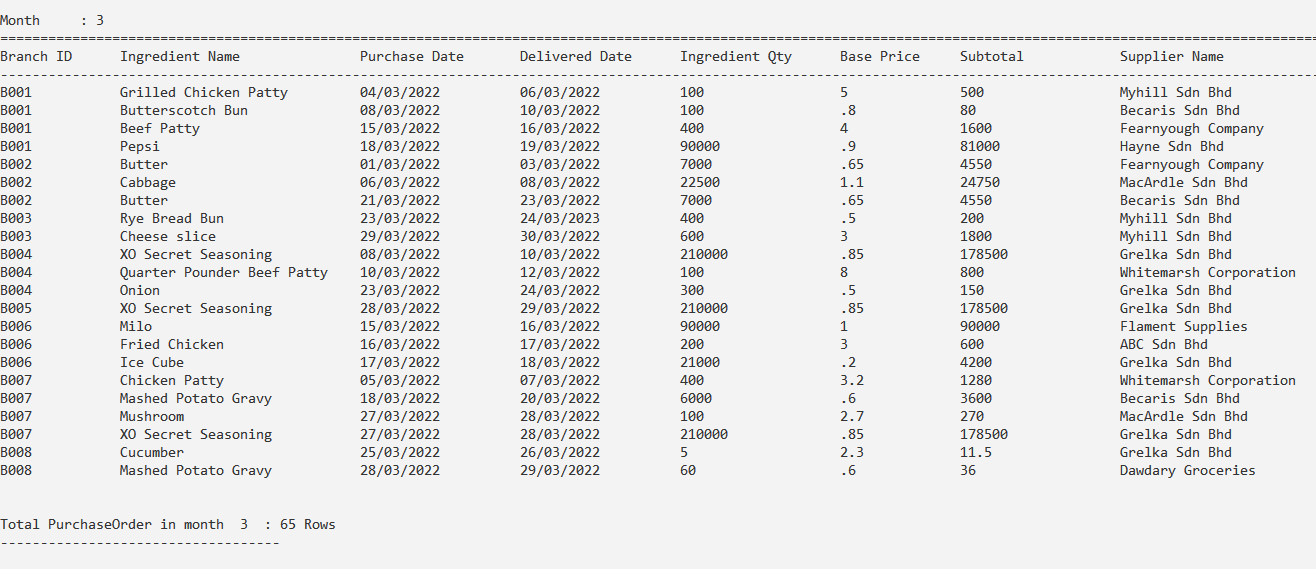
/

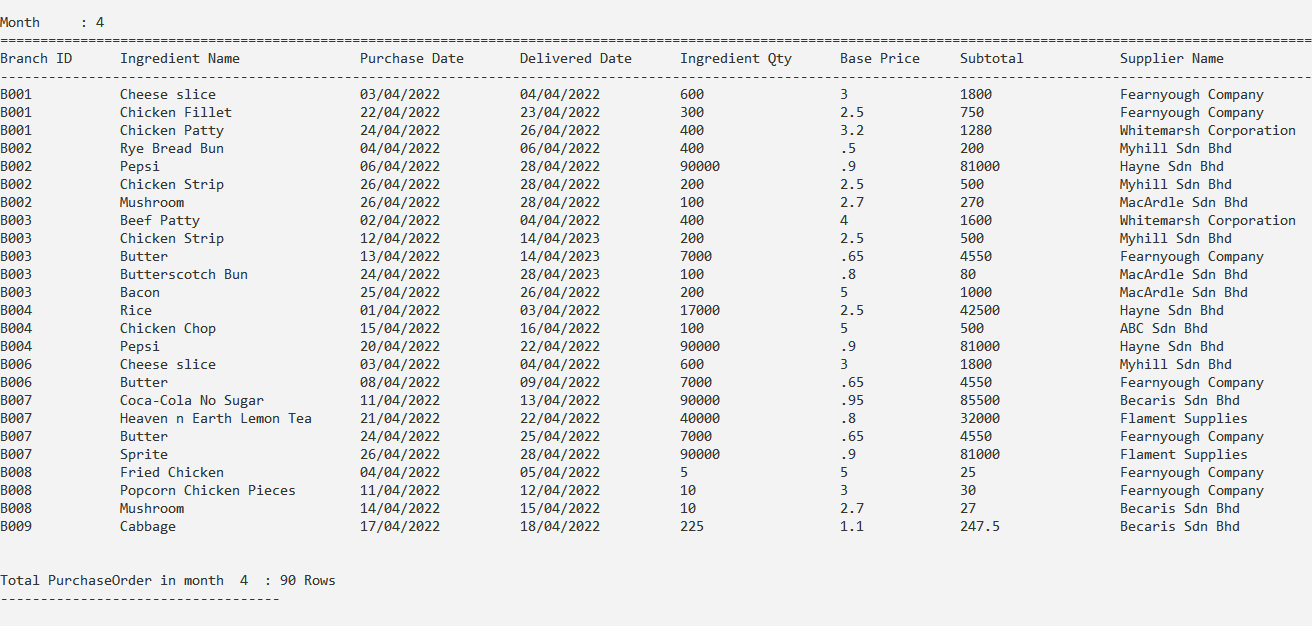
exec prc\_Total\_Purchase\_each\_Month(2022)

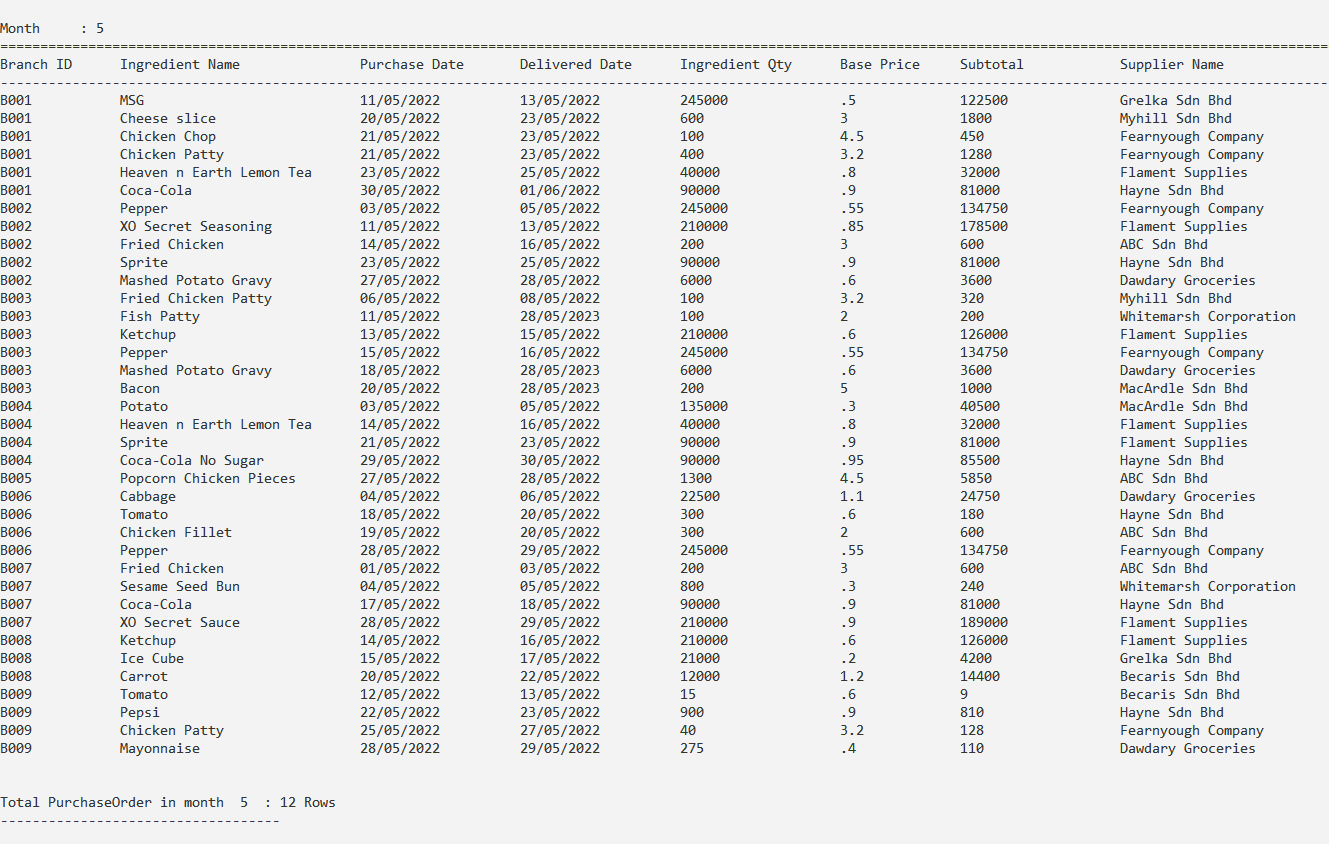
**Sample Output:**

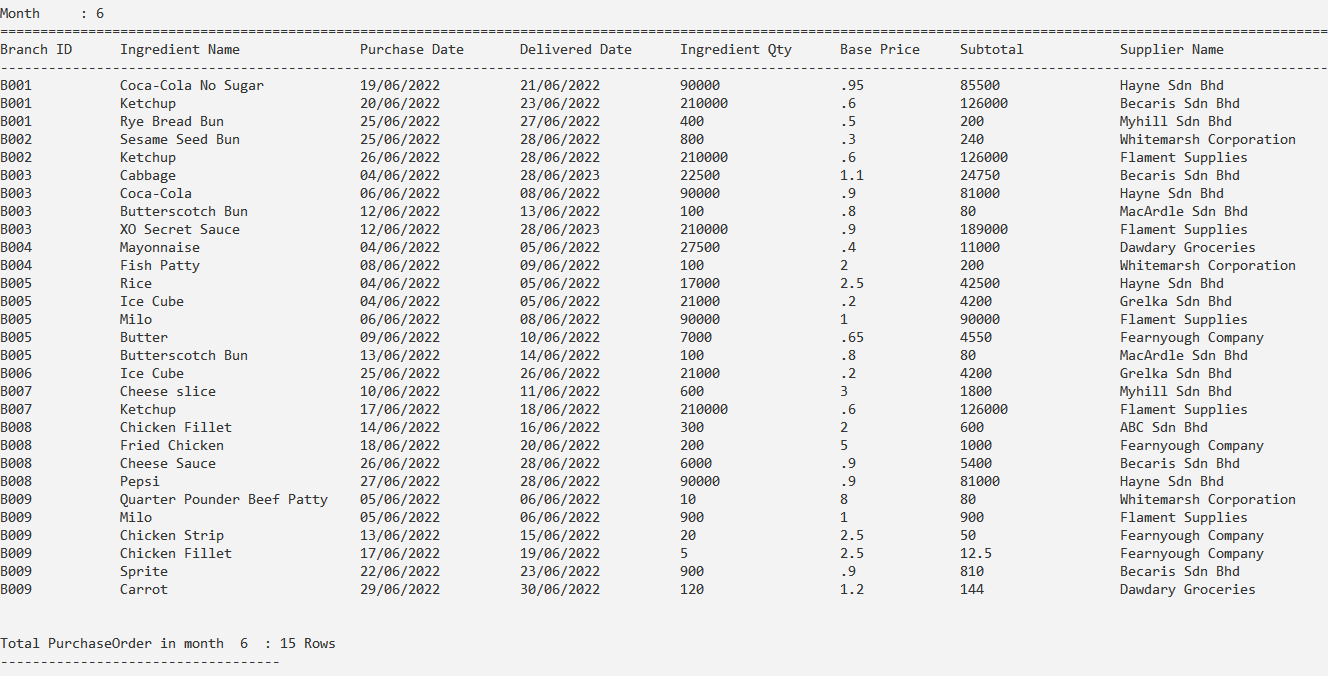


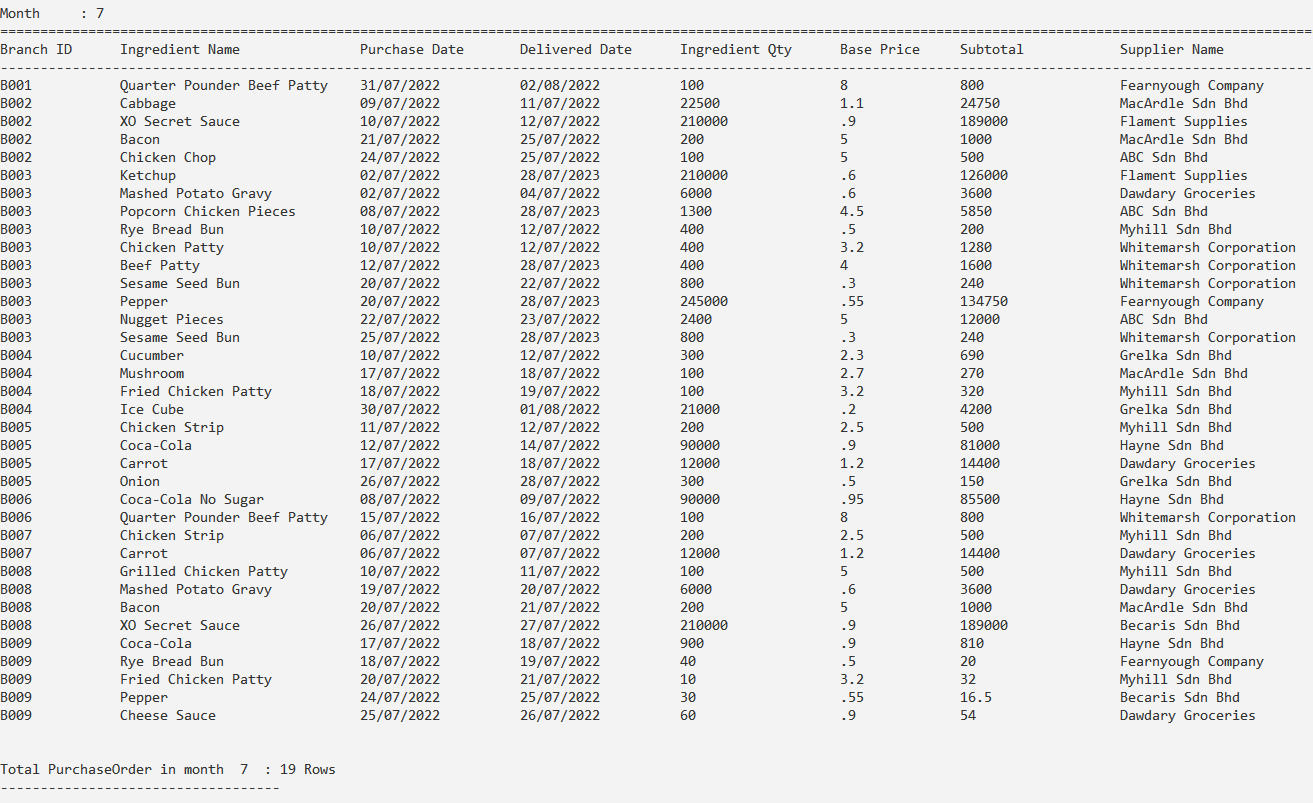




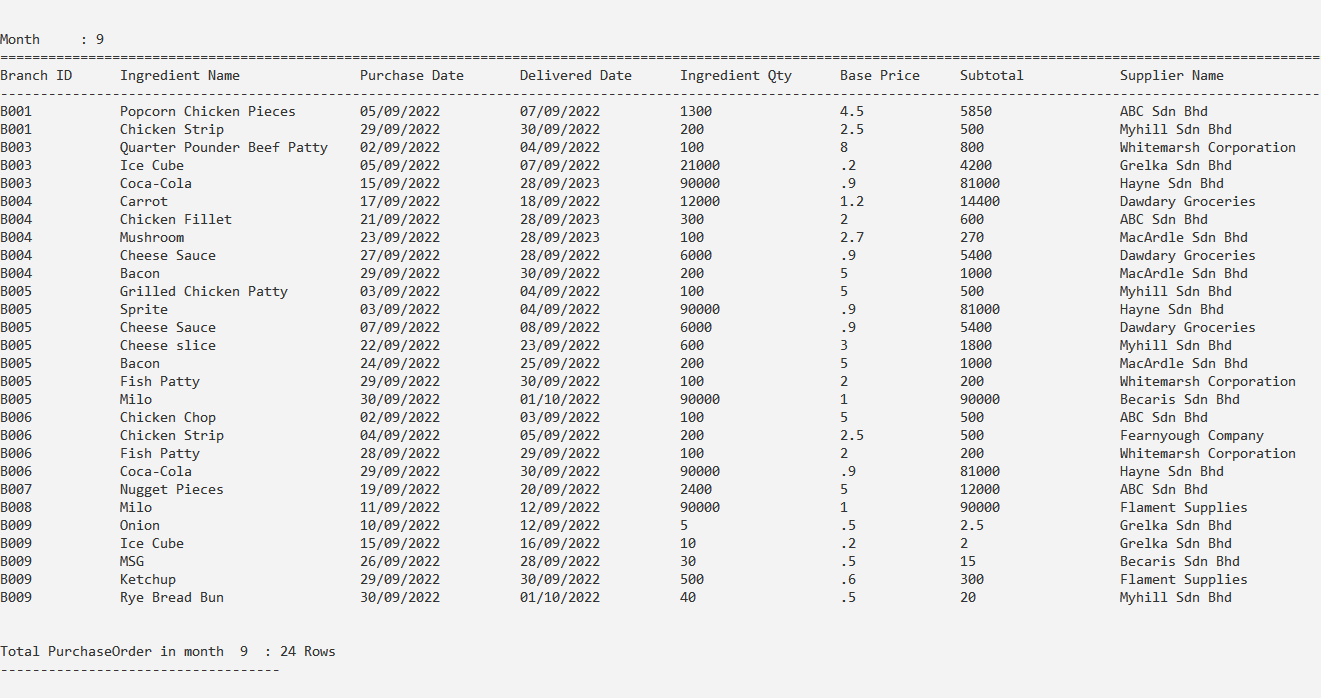


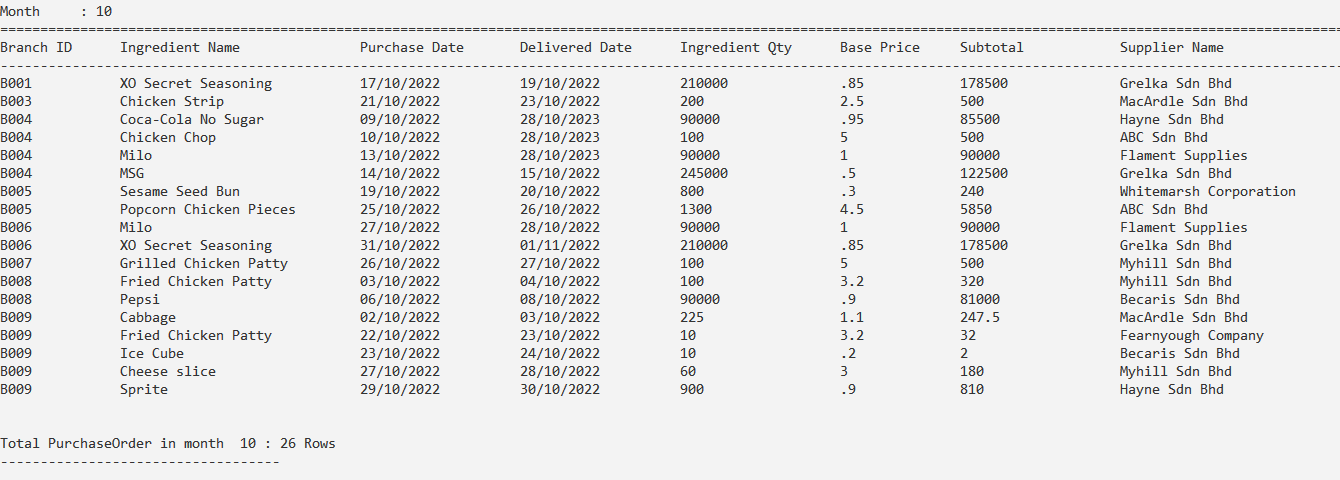


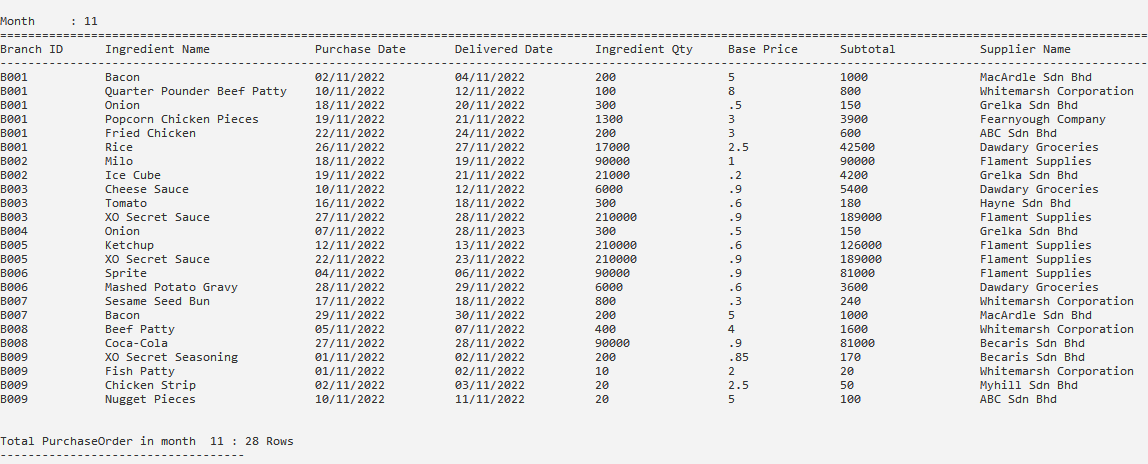


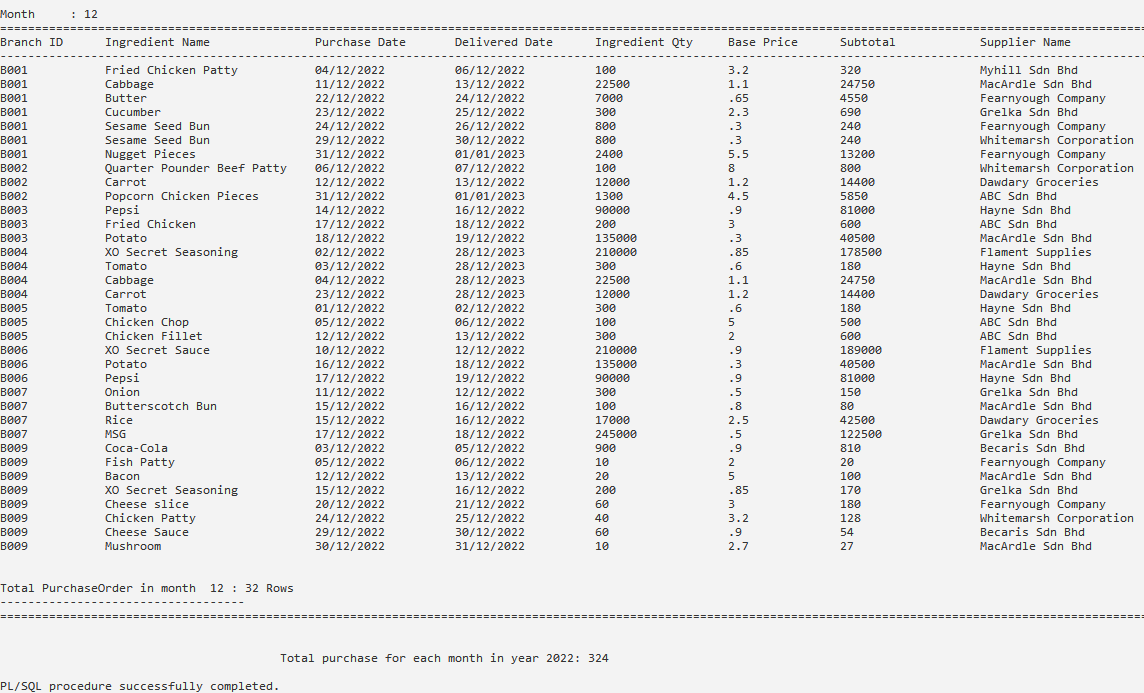












**4.3.8 Report 2: Report on Number of times delivery company process the order(On-Demand)**

**Purpose: The purpose of this report is to gather information about before and after 1pm deliveries made by a particular delivery company ('D001') and the associated details such as the branch, sale order dates, menu items, quantities, and the person who collected the order. The report helps in analyzing the efficiency of 'D001' in handling deliveries during different times of the day. The report segregates deliveries made before and after 1 PM. This allows the company to assess if there are any performance differences between morning and afternoon deliveries.**

**Report code:**

set linesize 1000

CREATE OR REPLACE PROCEDURE RPT\_DileveryO\_TAKEN(IN\_DeliveryCompanyID DeliveryCompany.DeliveryCompanyID%TYPE) IS

totalMorOrd NUMBER(3);

totalAftOrd NUMBER(3);

numberOfOrder NUMBER(3);

durationOrdMor NUMBER(3);

durationOrdAft NUMBER(3);

v\_DeliveryCompanyID DeliveryCompany.DeliveryCompanyID%TYPE;

v\_DeliveryCompanyName DeliveryCompany.DeliveryCompanyName%TYPE;

CURSOR Delivery\_Morn\_CURSOR IS

SELECT

DC.DeliveryCompanyName,

B.BranchName,

DOI.DelCity,

SO.SaleOrdDate,

SO.DateTimeCompleted,

M.MenuDesc,

SOI.Qty,

DOI.CollectedBy

FROM

DeliveryCompany DC, branch b, DeliveryOrderInfo DOI, SaleOrder SO, SaleOrderItem SOI, Menu M

where

DC.DeliveryCompanyID = DOI.DeliveryCompanyID

and

DOI.SaleOrdID = SO.SaleOrdID

and

SO.SaleOrdID = SOI.SaleOrdID

and

SOI.MenuID = M.MenuID

and

SO.BranchID = B.BranchID

and

TO\_CHAR(SO.SaleOrdDate, 'HH24:MI') < '13:00'

and

DOI.DeliveryCompanyID = IN\_DeliveryCompanyID

ORDER BY SO.SaleOrdDate;

CURSOR Delivery\_Aft\_CURSOR IS

SELECT

DC.DeliveryCompanyName,

B.BranchName,

DOI.DelCity,

SO.SaleOrdDate,

SO.DateTimeCompleted,

M.MenuDesc,

SOI.Qty,

DOI.CollectedBy

FROM

DeliveryCompany DC, branch b, DeliveryOrderInfo DOI, SaleOrder SO, SaleOrderItem SOI, Menu M

where

DC.DeliveryCompanyID = DOI.DeliveryCompanyID

and

DOI.SaleOrdID = SO.SaleOrdID

and

SO.SaleOrdID = SOI.SaleOrdID

and

SOI.MenuID = M.MenuID

and

SO.BranchID = B.BranchID

and

TO\_CHAR(SO.SaleOrdDate, 'HH24:MI') > '13:00'

and

DOI.DeliveryCompanyID = IN\_DeliveryCompanyID

ORDER BY SO.SaleOrdDate;

BEGIN

totalMorOrd := 0;

totalAftOrd:= 0;

numberOfOrder := 0;

SELECT DeliveryCompanyID, DeliveryCompanyName

INTO v\_DeliveryCompanyID, V\_DeliveryCompanyName

FROM DeliveryCompany

WHERE DeliveryCompanyID = IN\_DeliveryCompanyID;

DBMS\_OUTPUT.put\_line(CHR(10)||'Delivery Company''s Order');

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*',60,'\*'));

DBMS\_OUTPUT.put\_line(RPAD('DeliveryCompany ID',20,' ')|| ': '||v\_DeliveryCompanyID);

DBMS\_OUTPUT.put\_line(RPAD('DeliveryCompany Name',20,' ')|| ': '||V\_DeliveryCompanyName);

DBMS\_OUTPUT.PUT\_LINE(RPAD('\*',60,'\*'));

DBMS\_OUTPUT.PUT\_LINE(CHR(10));

DBMS\_OUTPUT.PUT\_LINE('Order Before 13 pm');

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',160,'='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Delivery Company Name',15,' ')||'| '||RPAD('Branch Name',15,' ')||'| '||RPAD('City',15,' ')||'| '||RPAD('Order Date Time',18,' ')||'| '||RPAD('Complete Date Time',20,' ')||'| '||RPAD('Menu Item Name',22,' ')||'| '||RPAD('Quantity',10,' ')||'| '||RPAD('Collect By',15,' '));

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',160,'='));

FOR deliveryC\_Ord IN Delivery\_Morn\_CURSOR LOOP

durationOrdMor := deliveryC\_Ord.DateTimeCompleted - deliveryC\_Ord.SaleOrdDate;

DBMS\_OUTPUT.PUT\_LINE(RPAD(deliveryC\_Ord.DeliveryCompanyName,15,' ')||'| '||RPAD(deliveryC\_Ord.BranchName,15,' ')||'| '||RPAD(deliveryC\_Ord.DelCity,15,' ')||'| '||RPAD(deliveryC\_Ord.SaleOrdDate,18,' ')||'| '||RPAD(deliveryC\_Ord.DateTimeCompleted,20,' ')||'| '||RPAD(deliveryC\_Ord.MenuDesc,22,' ')||'| '||RPAD(deliveryC\_Ord.Qty,10,' ')||'| '||RPAD(deliveryC\_Ord.CollectedBy,15,' '));

totalMorOrd := totalMorOrd + 1;

numberOfOrder:= numberOfOrder + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',160,'='));

DBMS\_OUTPUT.PUT\_LINE('Total Number of Delivery Order before 13 pm = '||totalMorOrd);

DBMS\_OUTPUT.PUT\_LINE(CHR(10));

DBMS\_OUTPUT.PUT\_LINE('Order After 13 pm');

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',160,'='));

DBMS\_OUTPUT.PUT\_LINE(RPAD('Delivery Company Name',15,' ')||'| '||RPAD('Branch Name',15,' ')||'| '||RPAD('City',15,' ')||'| '||RPAD('Order Date Time',18,' ')||'| '||RPAD('Complete Date Time',20,' ')||'| '||RPAD('Menu Item Name',28,' ')||'| '||RPAD('Quantity',10,' ')||'| '||RPAD('Collect By',15,' '));

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',160,'='));

FOR deliveryC\_Ord IN Delivery\_Aft\_CURSOR LOOP

durationOrdAft := deliveryC\_Ord.DateTimeCompleted - deliveryC\_Ord.SaleOrdDate;

DBMS\_OUTPUT.PUT\_LINE(RPAD(deliveryC\_Ord.DeliveryCompanyName,15,' ')||'| '||RPAD(deliveryC\_Ord.BranchName,15,' ')||'| '||RPAD(deliveryC\_Ord.DelCity,15,' ')||'| '||RPAD(deliveryC\_Ord.SaleOrdDate,18,' ')||'| '||RPAD(deliveryC\_Ord.DateTimeCompleted,20,' ')||'| '||RPAD(deliveryC\_Ord.MenuDesc,28,' ')||'| '||RPAD(deliveryC\_Ord.Qty,10,' ')||'| '||RPAD(deliveryC\_Ord.CollectedBy,15,' '));

totalAftOrd := totalAftOrd + 1;

numberOfOrder:= numberOfOrder + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(RPAD('=',160,'='));

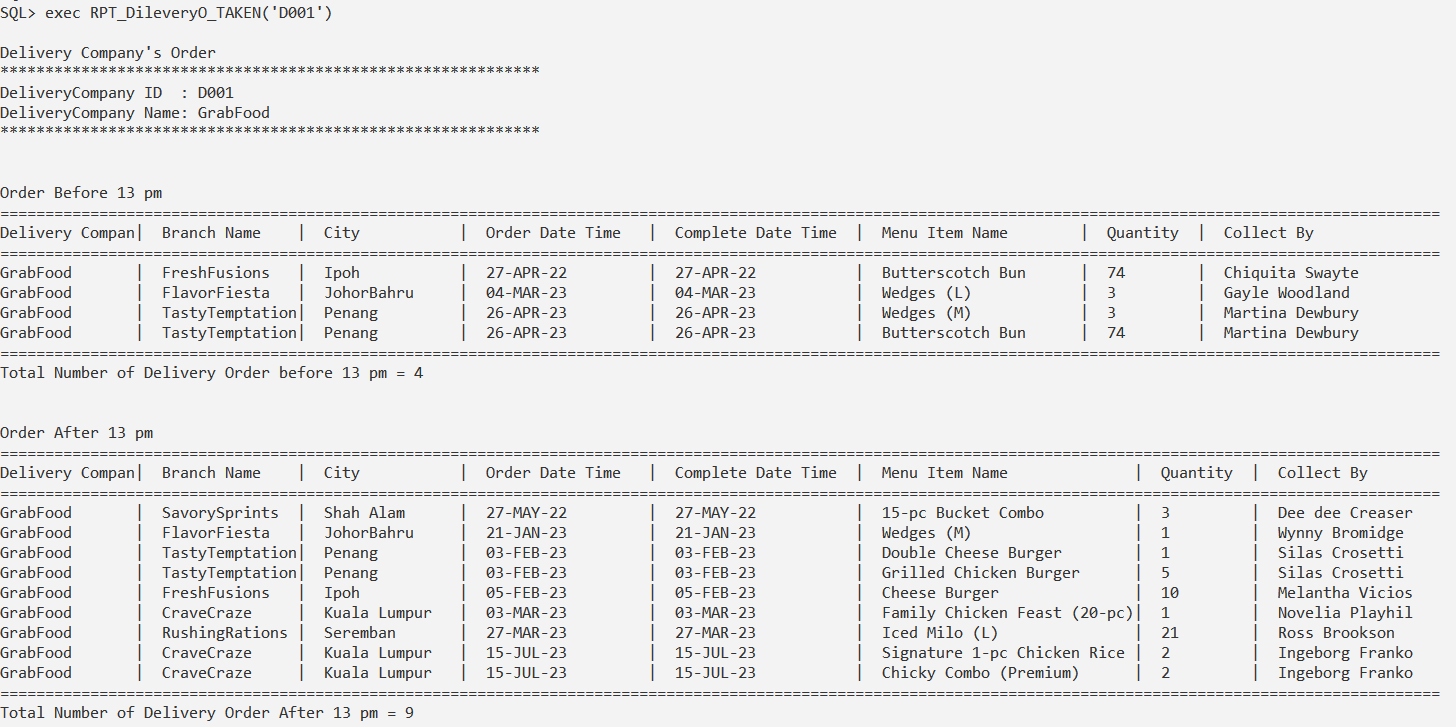
DBMS\_OUTPUT.PUT\_LINE(LPAD('Total Number of Delivery Order After 13 pm = ',50,' ')||totalAftOrd);

END;

/

exec RPT\_DileveryO\_TAKEN('D001')

**Sample Output:**



**4.4 (Thong Cheng How)**

**4.4.1 Query 1: Display all branch transactions and classify it by the highest transaction given the dates by the user input.**

**Purpose: The purpose of this query is to allow the end user to search through the transaction details made by the branch and to check the menu id that has been frequently called by the customers. By using this query, it helps the user to search by the dates where the menu is very popular and when it is not popular, this helps the management to make better decision making in terms of promotions and discounts to the customers in the fast food restaurant. The reason we search by date and time, we could make the decision and also when to promote the event. So in the future the revenue could increase for the restaurant.**

**Query code:**

CL SCR

SET SERVEROUTPUT ON

ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD/MM/YYYY HH24:MI';

SET LINESIZE 140

SET PAGESIZE 140

SET UNDERLINE =

column branchID format A12 heading 'Branch ID'

column MenuDesc for A35 heading 'Menu Description'

column Total\_Quantity format 999,999 heading 'Total Quantity'

column Total\_Grand\_Total format 999,999.99 heading 'Total Grand Total'

TTITLE LEFT 'Branch Profitability Report' SKIP 2 -

LEFT 'Current Date: ' \_DATE SKIP 2

ACCEPT sale\_order\_date\_int DATE FORMAT 'DD/MM/YYYY HH24:MI' PROMPT 'Please enter start date (DD/MM/YYYY HH:MI)->'

ACCEPT date\_completed\_int DATE FORMAT 'DD/MM/YYYY HH24:MI' PROMPT 'Please enter end date (DD/MM/YYYY HH:MI)->'

SELECT SO.BranchID, SI.MenuID, SUM(SI.Qty) AS Total\_Quantity, MenuDesc, SUM(GrandTotal) as Total\_Grand\_Total

FROM SaleOrder SO

JOIN SaleOrderItem SI ON SO.SaleOrdID = SI.SaleOrdID

JOIN Menu M ON M.MenuID = SI.MenuID

WHERE SO.SaleOrdDate >= TO\_DATE('&&sale\_order\_date\_int', 'DD/MM/YYYY HH24:MI')

AND SO.DateTimeCompleted <= TO\_DATE('&&date\_completed\_int', 'DD/MM/YYYY HH24:MI')

GROUP BY SO.BranchID, SI.MenuID, MenuDesc

ORDER BY SUM(GrandTotal) DESC, SUM(SI.Qty) DESC;

PROMPT Script Execution Completed

clear columns

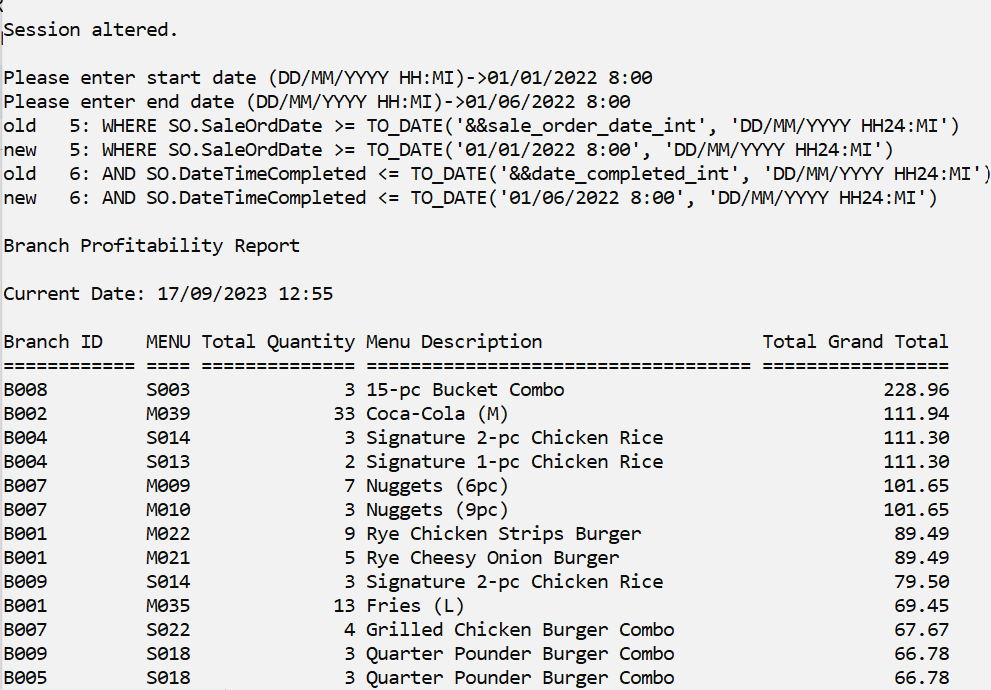
clear breaks

clear computes

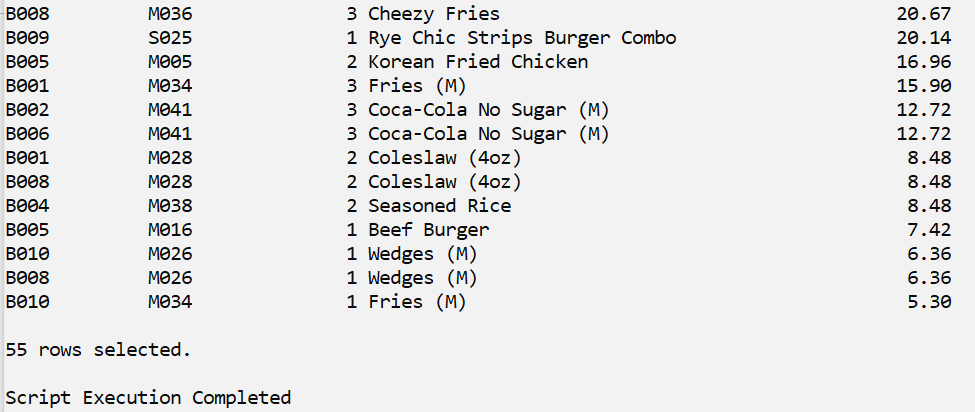
ttitle off

btitle off

**Sample Output:**







**4.4.2 Query 2: Display all the delivery information given the user input the states the user want to find out**

**Purpose: The purpose of this query is to find out the destination where the state is more famous for ordering or eating at the fast food restaurant. Knowing the location of the customer frequent purchase decisions, the management could open more stores near the palace to attract more customers to purchase the products offered by the fast food restaurant. By knowing the customers delivery order, the restaurant that are near by to the customers, the food preparation time for delivery or arriving to the venue to purchase the products are more convenient and faster.**

**Query code:**

CL SCR

SET SERVEROUTPUT ON

SET LINESIZE 140

SET PAGESIZE 140

SET UNDERLINE =

column delstate format A15 heading 'Delivery state'

column saleordid format A15 heading 'Sale Order ID'

column grandtotal format 999,999.99 heading 'Grand total'

column qty format 9999 heading 'Quantity'

column delcity format a15 heading 'Delivery City'

column delstreetaddress format a38 heading 'Delivery Street Address'

column delpostcode format 99999 heading 'Poster Code'

column datetimecompleted format a20 heading 'Date Time Completed'

TTITLE LEFT 'Query 2- Find the address of sale order id by state' SKIP 2-

LEFT 'Current date: ' \_DATE SKIP 2

CREATE OR REPLACE VIEW del\_state AS

SELECT DISTINCT DelState FROM DeliveryOrderInfo Order by DelState;

SELECT \* FROM del\_state;

ACCEPT DelState\_ipt CHAR PROMPT 'Please enter the state you want to find:';

SELECT SO.SaleOrdID, SO.GrandTotal, SI.MenuID, SI.Qty, D.DelCity, D.DelState, D.DelPostCode, D.DelStreetAddress, DateTimeCompleted

FROM SaleOrder SO

JOIN SaleOrderItem SI ON SO.SaleOrdID = SI.SaleOrdID

JOIN DeliveryOrderInfo D ON SO.SaleOrdID = D.SaleOrdID

WHERE D.DelState = UPPER('&DelState\_ipt')

ORDER BY SO.GrandTotal DESC;

PROMPT Script Execution Completed

clear columns

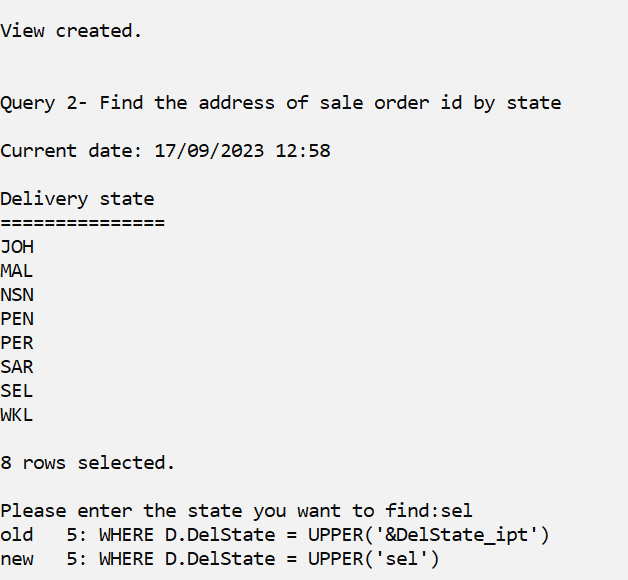
clear breaks

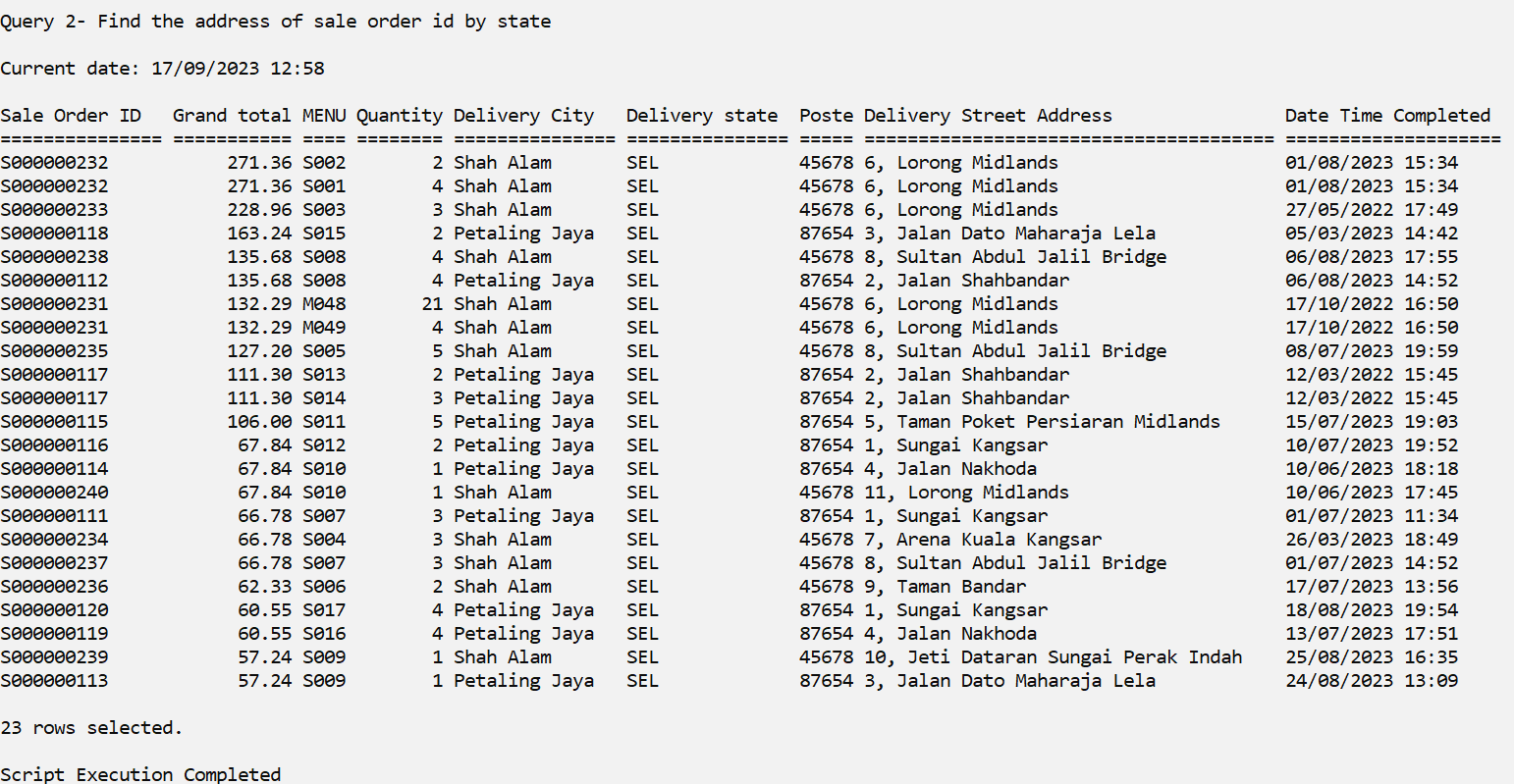
clear computes

ttitle off

btitle off

**Sample Output:**





**4.4.3 Trigger 1: To create a back up table for employee table to store data for the company**

**Trigger 1: to make an back-up employee table to keep track of the insert statements**

**Purpose: The purpose of this trigger is to create a back up table for employee. This is to save employee data if the data is suddenly corrupted or the data has been stolen by other users. By creating another table for back up, employee data will be kept safely, Employees also will not worry about the data being stolen because there is a back up if anything goes wrong.**

**Trigger code:**

CL SCR

DROP TABLE employees\_backup;

SET LINESIZE 200;

SET PAGESIZE 200;

CREATE TABLE employees\_backup AS SELECT \* FROM Employee WHERE 1=0;

ALTER TABLE employees\_backup ADD (action\_type VARCHAR2(10) DEFAULT 'N/A');

ALTER TABLE employees\_backup ADD (action\_date DATE DEFAULT SYSDATE);

--for insert

CREATE OR REPLACE TRIGGER employee\_after\_insert

AFTER INSERT ON Employee

FOR EACH ROW

BEGIN

INSERT INTO employees\_backup

SELECT e.\*, 'INSERT', SYSDATE

FROM Employee e

WHERE e.employeeID = :NEW.employeeID;

END;

/

--insert data

INSERT INTO employees\_backup (EmployeeID, Name, BirthDate, Gender, ContactNo, EmailAddr, BranchID, JobType, JobID, DateHired, ManagerID, action\_type, action\_date)

SELECT

'E00001',

'Terza Twist',

TO\_DATE('27/01/2004', 'DD/MM/YYYY'),

'F',

'8435524935',

'ttwist0@gizmodo.com',

'B001',

'FT',

'MAGR',

TO\_DATE('18/10/2016', 'DD/MM/YYYY'),

NULL,

'INSERT',

SYSDATE

FROM DUAL;

INSERT INTO employees\_backup (EmployeeID, Name, BirthDate, Gender, ContactNo, EmailAddr, BranchID, JobType, JobID, DateHired, ManagerID, action\_type, action\_date)

select

'E00002',

'Dunstan D''Alesco',

TO\_DATE('31/10/1987', 'DD/MM/YYYY'),

'M',

'9286577464',

'ddalesco1@salon.com',

'B001',

'FT',

'CASR',

TO\_DATE('23/10/2014', 'DD/MM/YYYY'),

'E00001',

'INSERT',

SYSDATE

FROM DUAL;

INSERT INTO employees\_backup (EmployeeID, Name, BirthDate, Gender, ContactNo, EmailAddr, BranchID, JobType, JobID, DateHired, ManagerID, action\_type, action\_date)

select

'E00003',

'Rudolfo Gaffon',

TO\_DATE('22/03/2003', 'DD/MM/YYYY'),

'M',

'8939540726',

'rgaffon2@salon.com',

'B001',

'FT',

'CHEF',

TO\_DATE('09/03/2012', 'DD/MM/YYYY'),

'E00001',

'INSERT',

SYSDATE

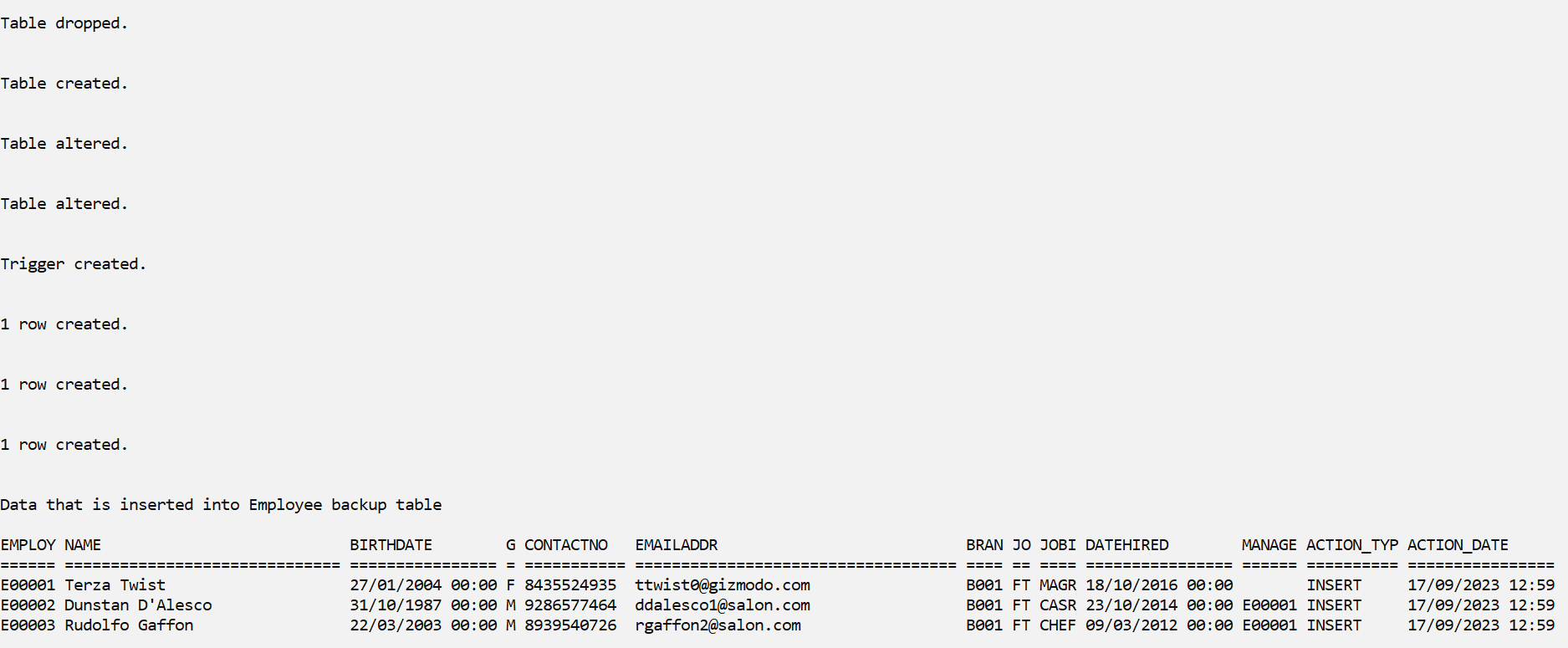
FROM DUAL;

--for inserted recrods (analysis)

TTITLE LEFT 'Data that is inserted into Employee backup table' SKIP 2

SELECT \* FROM employees\_backup WHERE action\_type = 'INSERT';

**Sample Output:**



**4.4.4 Trigger 2: To validate the insert statement for the employee back up table if the age is less than 16 years old**

**Purpose: The purpose of this trigger is to validate the insert statements in the backup table for employee, Validating the age for employee is crucial as companies has to follow the strict rules to avoid people that are working in a younger age, Because it's actually illegal to work as complied with the laws in Malaysia.**

**Trigger code:**

CL SCR

CREATE OR REPLACE TRIGGER trg\_validate\_employee\_data

BEFORE INSERT ON Employee

FOR EACH ROW

DECLARE

v\_yearsdiff NUMBER;

BEGIN

--Check if the email contains '@' and a domain

--IF NOT REGEXP\_LIKE(:NEW.EmailAddr, '^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$') THEN

--RAISE\_APPLICATION\_ERROR(-20001, 'Invalid email format');

--END IF;

-- Calculate the difference in years between the birth date and the current date

v\_yearsdiff := EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM :NEW.BirthDate);

-- Adjust the year difference if the birth date hasn't occurred yet for the current year

IF TO\_DATE(TO\_CHAR(EXTRACT(MONTH FROM :NEW.BirthDate)) || '/' || TO\_CHAR(EXTRACT(DAY FROM :NEW.BirthDate)), 'MM/DD') >

TO\_DATE(TO\_CHAR(EXTRACT(MONTH FROM SYSDATE)) || '/' || TO\_CHAR(EXTRACT(DAY FROM SYSDATE)), 'MM/DD') THEN

v\_yearsdiff := v\_yearsdiff - 1;

END IF;

-- Ensure the employee is at least 17 years old

IF v\_yearsdiff < 17 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee should be at least 16 years old');

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE;

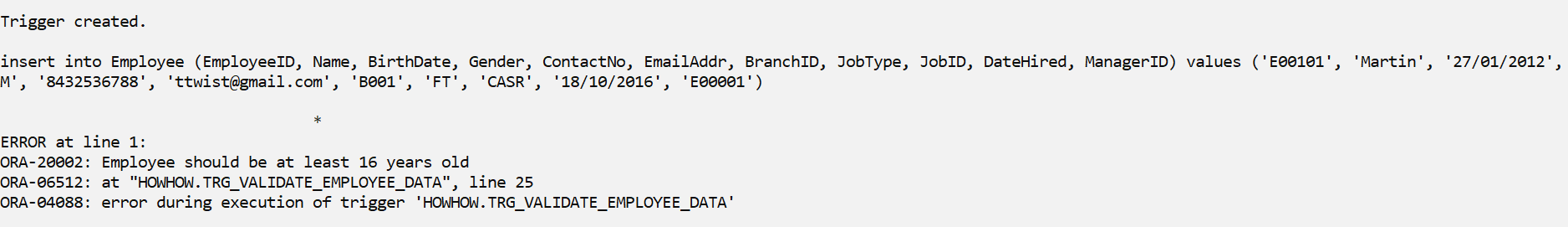
END trg\_validate\_employee\_data;

/

--TESTING THE VALIDATION

insert into Employee (EmployeeID, Name, BirthDate, Gender, ContactNo, EmailAddr, BranchID, JobType, JobID, DateHired, ManagerID) values ('E00101', 'Martin', '27/01/2012', 'M', '8432536788', 'ttwist@gmail.com', 'B001', 'FT', 'CASR', '18/10/2016', 'E00001');

**Sample Output:**



**4.4.5 Procedure 1: To update the sales tax and grand total in the sale order by getting the quantity and the base price from the sales order item table.**

**Purpose: The purpose of this procedure is to auto generate the sales tax and the grand total for each sales order in the system. The reason to use it is because by auto generating the sales tax and grand total, human error will likely be reduced as we humans may make careless mistakes when updating the sales tax and grand total calculations. So using the procedure to update the sales tax and grand total, the end user doesn't have to manually insert the records one by one. Another thing is that, if the quantity or the sales prices change, the procedure will also auto generate based on the quantity and price for the products, Hence reducing burden to the worker.**

**Procedure code:**

CL SCR

CREATE OR REPLACE Procedure UpdateSaleOrder(

p\_SaleOrdID IN SaleOrder.SaleOrdID%TYPE,

p\_SalesTax IN SaleOrder.SalesTax%TYPE DEFAULT NULL,

p\_GrandTotal IN SaleOrder.GrandTotal%TYPE DEFAULT NULL

) AS

v\_subtotal SaleOrder.GrandTotal%TYPE;

v\_salesTax SaleOrder.SalesTax%TYPE;

BEGIN

SELECT SUM(Subtotal) INTO v\_subtotal

FROM SaleOrderItem

WHERE SaleOrdID = p\_SaleOrdID;

v\_salesTax := v\_subtotal \* 0.06;

UPDATE SaleOrder

SET SalesTax = NVL(p\_SalesTax, v\_salesTax),

GrandTotal = v\_subtotal + NVL(p\_SalesTax, v\_salesTax)

WHERE SaleOrdID = p\_SaleOrdID;

COMMIT;

END UpdateSaleOrder;

/

BEGIN

UpdateSaleOrder(p\_SaleOrdID => 'S000000001');

UpdateSaleOrder(p\_SaleOrdID => 'S000000002');

UpdateSaleOrder(p\_SaleOrdID => 'S000000003');

UpdateSaleOrder(p\_SaleOrdID => 'S000000004');

UpdateSaleOrder(p\_SaleOrdID => 'S000000005');

UpdateSaleOrder(p\_SaleOrdID => 'S000000006');

UpdateSaleOrder(p\_SaleOrdID => 'S000000007');

UpdateSaleOrder(p\_SaleOrdID => 'S000000008');

UpdateSaleOrder(p\_SaleOrdID => 'S000000009');

UpdateSaleOrder(p\_SaleOrdID => 'S000000010');

UpdateSaleOrder(p\_SaleOrdID => 'S000000011');

UpdateSaleOrder(p\_SaleOrdID => 'S000000012');

UpdateSaleOrder(p\_SaleOrdID => 'S000000013');

UpdateSaleOrder(p\_SaleOrdID => 'S000000014');

UpdateSaleOrder(p\_SaleOrdID => 'S000000015');

UpdateSaleOrder(p\_SaleOrdID => 'S000000016');

UpdateSaleOrder(p\_SaleOrdID => 'S000000017');

UpdateSaleOrder(p\_SaleOrdID => 'S000000018');

UpdateSaleOrder(p\_SaleOrdID => 'S000000019');

UpdateSaleOrder(p\_SaleOrdID => 'S000000020');

UpdateSaleOrder(p\_SaleOrdID => 'S000000021');

UpdateSaleOrder(p\_SaleOrdID => 'S000000022');

UpdateSaleOrder(p\_SaleOrdID => 'S000000023');

UpdateSaleOrder(p\_SaleOrdID => 'S000000024');

UpdateSaleOrder(p\_SaleOrdID => 'S000000025');

UpdateSaleOrder(p\_SaleOrdID => 'S000000026');

UpdateSaleOrder(p\_SaleOrdID => 'S000000027');

UpdateSaleOrder(p\_SaleOrdID => 'S000000028');

UpdateSaleOrder(p\_SaleOrdID => 'S000000029');

UpdateSaleOrder(p\_SaleOrdID => 'S000000030');

UpdateSaleOrder(p\_SaleOrdID => 'S000000031');

UpdateSaleOrder(p\_SaleOrdID => 'S000000032');

UpdateSaleOrder(p\_SaleOrdID => 'S000000033');

UpdateSaleOrder(p\_SaleOrdID => 'S000000034');

UpdateSaleOrder(p\_SaleOrdID => 'S000000035');

UpdateSaleOrder(p\_SaleOrdID => 'S000000036');

UpdateSaleOrder(p\_SaleOrdID => 'S000000037');

UpdateSaleOrder(p\_SaleOrdID => 'S000000038');

UpdateSaleOrder(p\_SaleOrdID => 'S000000039');

UpdateSaleOrder(p\_SaleOrdID => 'S000000040');

UpdateSaleOrder(p\_SaleOrdID => 'S000000041');

UpdateSaleOrder(p\_SaleOrdID => 'S000000042');

UpdateSaleOrder(p\_SaleOrdID => 'S000000043');

UpdateSaleOrder(p\_SaleOrdID => 'S000000044');

UpdateSaleOrder(p\_SaleOrdID => 'S000000045');

UpdateSaleOrder(p\_SaleOrdID => 'S000000046');

UpdateSaleOrder(p\_SaleOrdID => 'S000000047');

UpdateSaleOrder(p\_SaleOrdID => 'S000000048');

UpdateSaleOrder(p\_SaleOrdID => 'S000000049');

UpdateSaleOrder(p\_SaleOrdID => 'S000000050');

UpdateSaleOrder(p\_SaleOrdID => 'S000000051');

UpdateSaleOrder(p\_SaleOrdID => 'S000000052');

UpdateSaleOrder(p\_SaleOrdID => 'S000000053');

UpdateSaleOrder(p\_SaleOrdID => 'S000000054');

UpdateSaleOrder(p\_SaleOrdID => 'S000000055');

UpdateSaleOrder(p\_SaleOrdID => 'S000000056');

UpdateSaleOrder(p\_SaleOrdID => 'S000000057');

UpdateSaleOrder(p\_SaleOrdID => 'S000000058');

UpdateSaleOrder(p\_SaleOrdID => 'S000000059');

UpdateSaleOrder(p\_SaleOrdID => 'S000000060');

UpdateSaleOrder(p\_SaleOrdID => 'S000000061');

UpdateSaleOrder(p\_SaleOrdID => 'S000000062');

UpdateSaleOrder(p\_SaleOrdID => 'S000000063');

UpdateSaleOrder(p\_SaleOrdID => 'S000000064');

UpdateSaleOrder(p\_SaleOrdID => 'S000000065');

UpdateSaleOrder(p\_SaleOrdID => 'S000000066');

UpdateSaleOrder(p\_SaleOrdID => 'S000000067');

UpdateSaleOrder(p\_SaleOrdID => 'S000000068');

UpdateSaleOrder(p\_SaleOrdID => 'S000000069');

UpdateSaleOrder(p\_SaleOrdID => 'S000000070');

UpdateSaleOrder(p\_SaleOrdID => 'S000000071');

UpdateSaleOrder(p\_SaleOrdID => 'S000000072');

UpdateSaleOrder(p\_SaleOrdID => 'S000000073');

UpdateSaleOrder(p\_SaleOrdID => 'S000000074');

UpdateSaleOrder(p\_SaleOrdID => 'S000000075');

UpdateSaleOrder(p\_SaleOrdID => 'S000000076');

UpdateSaleOrder(p\_SaleOrdID => 'S000000077');

UpdateSaleOrder(p\_SaleOrdID => 'S000000078');

UpdateSaleOrder(p\_SaleOrdID => 'S000000079');

UpdateSaleOrder(p\_SaleOrdID => 'S000000080');

UpdateSaleOrder(p\_SaleOrdID => 'S000000081');

UpdateSaleOrder(p\_SaleOrdID => 'S000000082');

UpdateSaleOrder(p\_SaleOrdID => 'S000000083');

UpdateSaleOrder(p\_SaleOrdID => 'S000000084');

UpdateSaleOrder(p\_SaleOrdID => 'S000000085');

UpdateSaleOrder(p\_SaleOrdID => 'S000000086');

UpdateSaleOrder(p\_SaleOrdID => 'S000000087');

UpdateSaleOrder(p\_SaleOrdID => 'S000000088');

UpdateSaleOrder(p\_SaleOrdID => 'S000000089');

UpdateSaleOrder(p\_SaleOrdID => 'S000000090');

UpdateSaleOrder(p\_SaleOrdID => 'S000000091');

UpdateSaleOrder(p\_SaleOrdID => 'S000000092');

UpdateSaleOrder(p\_SaleOrdID => 'S000000093');

UpdateSaleOrder(p\_SaleOrdID => 'S000000094');

UpdateSaleOrder(p\_SaleOrdID => 'S000000095');

UpdateSaleOrder(p\_SaleOrdID => 'S000000096');

UpdateSaleOrder(p\_SaleOrdID => 'S000000097');

UpdateSaleOrder(p\_SaleOrdID => 'S000000098');

UpdateSaleOrder(p\_SaleOrdID => 'S000000099');

UpdateSaleOrder(p\_SaleOrdID => 'S000000100');

UpdateSaleOrder(p\_SaleOrdID => 'S000000101');

UpdateSaleOrder(p\_SaleOrdID => 'S000000102');

UpdateSaleOrder(p\_SaleOrdID => 'S000000103');

UpdateSaleOrder(p\_SaleOrdID => 'S000000104');

UpdateSaleOrder(p\_SaleOrdID => 'S000000105');

UpdateSaleOrder(p\_SaleOrdID => 'S000000106');

UpdateSaleOrder(p\_SaleOrdID => 'S000000107');

UpdateSaleOrder(p\_SaleOrdID => 'S000000108');

UpdateSaleOrder(p\_SaleOrdID => 'S000000109');

UpdateSaleOrder(p\_SaleOrdID => 'S000000110');

UpdateSaleOrder(p\_SaleOrdID => 'S000000111');

UpdateSaleOrder(p\_SaleOrdID => 'S000000112');

UpdateSaleOrder(p\_SaleOrdID => 'S000000113');

UpdateSaleOrder(p\_SaleOrdID => 'S000000114');

UpdateSaleOrder(p\_SaleOrdID => 'S000000115');

UpdateSaleOrder(p\_SaleOrdID => 'S000000116');

UpdateSaleOrder(p\_SaleOrdID => 'S000000117');

UpdateSaleOrder(p\_SaleOrdID => 'S000000118');

UpdateSaleOrder(p\_SaleOrdID => 'S000000119');

UpdateSaleOrder(p\_SaleOrdID => 'S000000120');

UpdateSaleOrder(p\_SaleOrdID => 'S000000121');

UpdateSaleOrder(p\_SaleOrdID => 'S000000122');

UpdateSaleOrder(p\_SaleOrdID => 'S000000123');

UpdateSaleOrder(p\_SaleOrdID => 'S000000124');

UpdateSaleOrder(p\_SaleOrdID => 'S000000125');

UpdateSaleOrder(p\_SaleOrdID => 'S000000126');

UpdateSaleOrder(p\_SaleOrdID => 'S000000127');

UpdateSaleOrder(p\_SaleOrdID => 'S000000128');

UpdateSaleOrder(p\_SaleOrdID => 'S000000129');

UpdateSaleOrder(p\_SaleOrdID => 'S000000130');

UpdateSaleOrder(p\_SaleOrdID => 'S000000131');

UpdateSaleOrder(p\_SaleOrdID => 'S000000132');

UpdateSaleOrder(p\_SaleOrdID => 'S000000133');

UpdateSaleOrder(p\_SaleOrdID => 'S000000134');

UpdateSaleOrder(p\_SaleOrdID => 'S000000135');

UpdateSaleOrder(p\_SaleOrdID => 'S000000136');

UpdateSaleOrder(p\_SaleOrdID => 'S000000137');

UpdateSaleOrder(p\_SaleOrdID => 'S000000138');

UpdateSaleOrder(p\_SaleOrdID => 'S000000139');

UpdateSaleOrder(p\_SaleOrdID => 'S000000140');

UpdateSaleOrder(p\_SaleOrdID => 'S000000141');

UpdateSaleOrder(p\_SaleOrdID => 'S000000142');

UpdateSaleOrder(p\_SaleOrdID => 'S000000143');

UpdateSaleOrder(p\_SaleOrdID => 'S000000144');

UpdateSaleOrder(p\_SaleOrdID => 'S000000145');

UpdateSaleOrder(p\_SaleOrdID => 'S000000146');

UpdateSaleOrder(p\_SaleOrdID => 'S000000147');

UpdateSaleOrder(p\_SaleOrdID => 'S000000148');

UpdateSaleOrder(p\_SaleOrdID => 'S000000149');

UpdateSaleOrder(p\_SaleOrdID => 'S000000150');

UpdateSaleOrder(p\_SaleOrdID => 'S000000151');

UpdateSaleOrder(p\_SaleOrdID => 'S000000152');

UpdateSaleOrder(p\_SaleOrdID => 'S000000153');

UpdateSaleOrder(p\_SaleOrdID => 'S000000154');

UpdateSaleOrder(p\_SaleOrdID => 'S000000155');

UpdateSaleOrder(p\_SaleOrdID => 'S000000156');

UpdateSaleOrder(p\_SaleOrdID => 'S000000157');

UpdateSaleOrder(p\_SaleOrdID => 'S000000158');

UpdateSaleOrder(p\_SaleOrdID => 'S000000159');

UpdateSaleOrder(p\_SaleOrdID => 'S000000160');

UpdateSaleOrder(p\_SaleOrdID => 'S000000161');

UpdateSaleOrder(p\_SaleOrdID => 'S000000162');

UpdateSaleOrder(p\_SaleOrdID => 'S000000163');

UpdateSaleOrder(p\_SaleOrdID => 'S000000164');

UpdateSaleOrder(p\_SaleOrdID => 'S000000165');

UpdateSaleOrder(p\_SaleOrdID => 'S000000166');

UpdateSaleOrder(p\_SaleOrdID => 'S000000167');

UpdateSaleOrder(p\_SaleOrdID => 'S000000168');

UpdateSaleOrder(p\_SaleOrdID => 'S000000169');

UpdateSaleOrder(p\_SaleOrdID => 'S000000170');

UpdateSaleOrder(p\_SaleOrdID => 'S000000171');

UpdateSaleOrder(p\_SaleOrdID => 'S000000172');

UpdateSaleOrder(p\_SaleOrdID => 'S000000173');

UpdateSaleOrder(p\_SaleOrdID => 'S000000174');

UpdateSaleOrder(p\_SaleOrdID => 'S000000175');

UpdateSaleOrder(p\_SaleOrdID => 'S000000176');

UpdateSaleOrder(p\_SaleOrdID => 'S000000177');

UpdateSaleOrder(p\_SaleOrdID => 'S000000178');

UpdateSaleOrder(p\_SaleOrdID => 'S000000179');

UpdateSaleOrder(p\_SaleOrdID => 'S000000180');

UpdateSaleOrder(p\_SaleOrdID => 'S000000181');

UpdateSaleOrder(p\_SaleOrdID => 'S000000182');

UpdateSaleOrder(p\_SaleOrdID => 'S000000183');

UpdateSaleOrder(p\_SaleOrdID => 'S000000184');

UpdateSaleOrder(p\_SaleOrdID => 'S000000185');

UpdateSaleOrder(p\_SaleOrdID => 'S000000186');

UpdateSaleOrder(p\_SaleOrdID => 'S000000187');

UpdateSaleOrder(p\_SaleOrdID => 'S000000188');

UpdateSaleOrder(p\_SaleOrdID => 'S000000189');

UpdateSaleOrder(p\_SaleOrdID => 'S000000190');

UpdateSaleOrder(p\_SaleOrdID => 'S000000191');

UpdateSaleOrder(p\_SaleOrdID => 'S000000192');

UpdateSaleOrder(p\_SaleOrdID => 'S000000193');

UpdateSaleOrder(p\_SaleOrdID => 'S000000194');

UpdateSaleOrder(p\_SaleOrdID => 'S000000195');

UpdateSaleOrder(p\_SaleOrdID => 'S000000196');

UpdateSaleOrder(p\_SaleOrdID => 'S000000197');

UpdateSaleOrder(p\_SaleOrdID => 'S000000198');

UpdateSaleOrder(p\_SaleOrdID => 'S000000199');

UpdateSaleOrder(p\_SaleOrdID => 'S000000200');

UpdateSaleOrder(p\_SaleOrdID => 'S000000201');

UpdateSaleOrder(p\_SaleOrdID => 'S000000202');

UpdateSaleOrder(p\_SaleOrdID => 'S000000203');

UpdateSaleOrder(p\_SaleOrdID => 'S000000204');

UpdateSaleOrder(p\_SaleOrdID => 'S000000205');

UpdateSaleOrder(p\_SaleOrdID => 'S000000206');

UpdateSaleOrder(p\_SaleOrdID => 'S000000207');

UpdateSaleOrder(p\_SaleOrdID => 'S000000208');

UpdateSaleOrder(p\_SaleOrdID => 'S000000209');

UpdateSaleOrder(p\_SaleOrdID => 'S000000210');

UpdateSaleOrder(p\_SaleOrdID => 'S000000211');

UpdateSaleOrder(p\_SaleOrdID => 'S000000212');

UpdateSaleOrder(p\_SaleOrdID => 'S000000213');

UpdateSaleOrder(p\_SaleOrdID => 'S000000214');

UpdateSaleOrder(p\_SaleOrdID => 'S000000215');

UpdateSaleOrder(p\_SaleOrdID => 'S000000216');

UpdateSaleOrder(p\_SaleOrdID => 'S000000217');

UpdateSaleOrder(p\_SaleOrdID => 'S000000218');

UpdateSaleOrder(p\_SaleOrdID => 'S000000219');

UpdateSaleOrder(p\_SaleOrdID => 'S000000220');

UpdateSaleOrder(p\_SaleOrdID => 'S000000221');

UpdateSaleOrder(p\_SaleOrdID => 'S000000222');

UpdateSaleOrder(p\_SaleOrdID => 'S000000223');

UpdateSaleOrder(p\_SaleOrdID => 'S000000224');

UpdateSaleOrder(p\_SaleOrdID => 'S000000225');

UpdateSaleOrder(p\_SaleOrdID => 'S000000226');

UpdateSaleOrder(p\_SaleOrdID => 'S000000227');

UpdateSaleOrder(p\_SaleOrdID => 'S000000228');

UpdateSaleOrder(p\_SaleOrdID => 'S000000229');

UpdateSaleOrder(p\_SaleOrdID => 'S000000230');

UpdateSaleOrder(p\_SaleOrdID => 'S000000231');

UpdateSaleOrder(p\_SaleOrdID => 'S000000232');

UpdateSaleOrder(p\_SaleOrdID => 'S000000233');

UpdateSaleOrder(p\_SaleOrdID => 'S000000234');

UpdateSaleOrder(p\_SaleOrdID => 'S000000235');

UpdateSaleOrder(p\_SaleOrdID => 'S000000236');

UpdateSaleOrder(p\_SaleOrdID => 'S000000237');

UpdateSaleOrder(p\_SaleOrdID => 'S000000238');

UpdateSaleOrder(p\_SaleOrdID => 'S000000239');

UpdateSaleOrder(p\_SaleOrdID => 'S000000240');

UpdateSaleOrder(p\_SaleOrdID => 'S000000241');

UpdateSaleOrder(p\_SaleOrdID => 'S000000242');

UpdateSaleOrder(p\_SaleOrdID => 'S000000243');

UpdateSaleOrder(p\_SaleOrdID => 'S000000244');

UpdateSaleOrder(p\_SaleOrdID => 'S000000245');

UpdateSaleOrder(p\_SaleOrdID => 'S000000246');

UpdateSaleOrder(p\_SaleOrdID => 'S000000247');

UpdateSaleOrder(p\_SaleOrdID => 'S000000248');

UpdateSaleOrder(p\_SaleOrdID => 'S000000249');

UpdateSaleOrder(p\_SaleOrdID => 'S000000250');

UpdateSaleOrder(p\_SaleOrdID => 'S000000251');

UpdateSaleOrder(p\_SaleOrdID => 'S000000252');

UpdateSaleOrder(p\_SaleOrdID => 'S000000253');

UpdateSaleOrder(p\_SaleOrdID => 'S000000254');

UpdateSaleOrder(p\_SaleOrdID => 'S000000255');

UpdateSaleOrder(p\_SaleOrdID => 'S000000256');

UpdateSaleOrder(p\_SaleOrdID => 'S000000257');

UpdateSaleOrder(p\_SaleOrdID => 'S000000258');

UpdateSaleOrder(p\_SaleOrdID => 'S000000259');

UpdateSaleOrder(p\_SaleOrdID => 'S000000260');

UpdateSaleOrder(p\_SaleOrdID => 'S000000261');

UpdateSaleOrder(p\_SaleOrdID => 'S000000262');

UpdateSaleOrder(p\_SaleOrdID => 'S000000263');

UpdateSaleOrder(p\_SaleOrdID => 'S000000264');

UpdateSaleOrder(p\_SaleOrdID => 'S000000265');

UpdateSaleOrder(p\_SaleOrdID => 'S000000266');

UpdateSaleOrder(p\_SaleOrdID => 'S000000267');

UpdateSaleOrder(p\_SaleOrdID => 'S000000268');

UpdateSaleOrder(p\_SaleOrdID => 'S000000269');

UpdateSaleOrder(p\_SaleOrdID => 'S000000270');

UpdateSaleOrder(p\_SaleOrdID => 'S000000271');

UpdateSaleOrder(p\_SaleOrdID => 'S000000272');

UpdateSaleOrder(p\_SaleOrdID => 'S000000273');

UpdateSaleOrder(p\_SaleOrdID => 'S000000274');

UpdateSaleOrder(p\_SaleOrdID => 'S000000275');

UpdateSaleOrder(p\_SaleOrdID => 'S000000276');

UpdateSaleOrder(p\_SaleOrdID => 'S000000277');

UpdateSaleOrder(p\_SaleOrdID => 'S000000278');

UpdateSaleOrder(p\_SaleOrdID => 'S000000279');

UpdateSaleOrder(p\_SaleOrdID => 'S000000280');

UpdateSaleOrder(p\_SaleOrdID => 'S000000281');

UpdateSaleOrder(p\_SaleOrdID => 'S000000282');

UpdateSaleOrder(p\_SaleOrdID => 'S000000283');

UpdateSaleOrder(p\_SaleOrdID => 'S000000284');

UpdateSaleOrder(p\_SaleOrdID => 'S000000285');

UpdateSaleOrder(p\_SaleOrdID => 'S000000286');

UpdateSaleOrder(p\_SaleOrdID => 'S000000287');

UpdateSaleOrder(p\_SaleOrdID => 'S000000288');

UpdateSaleOrder(p\_SaleOrdID => 'S000000289');

UpdateSaleOrder(p\_SaleOrdID => 'S000000290');

UpdateSaleOrder(p\_SaleOrdID => 'S000000291');

UpdateSaleOrder(p\_SaleOrdID => 'S000000292');

UpdateSaleOrder(p\_SaleOrdID => 'S000000293');

UpdateSaleOrder(p\_SaleOrdID => 'S000000294');

UpdateSaleOrder(p\_SaleOrdID => 'S000000295');

UpdateSaleOrder(p\_SaleOrdID => 'S000000296');

UpdateSaleOrder(p\_SaleOrdID => 'S000000297');

UpdateSaleOrder(p\_SaleOrdID => 'S000000298');

UpdateSaleOrder(p\_SaleOrdID => 'S000000299');

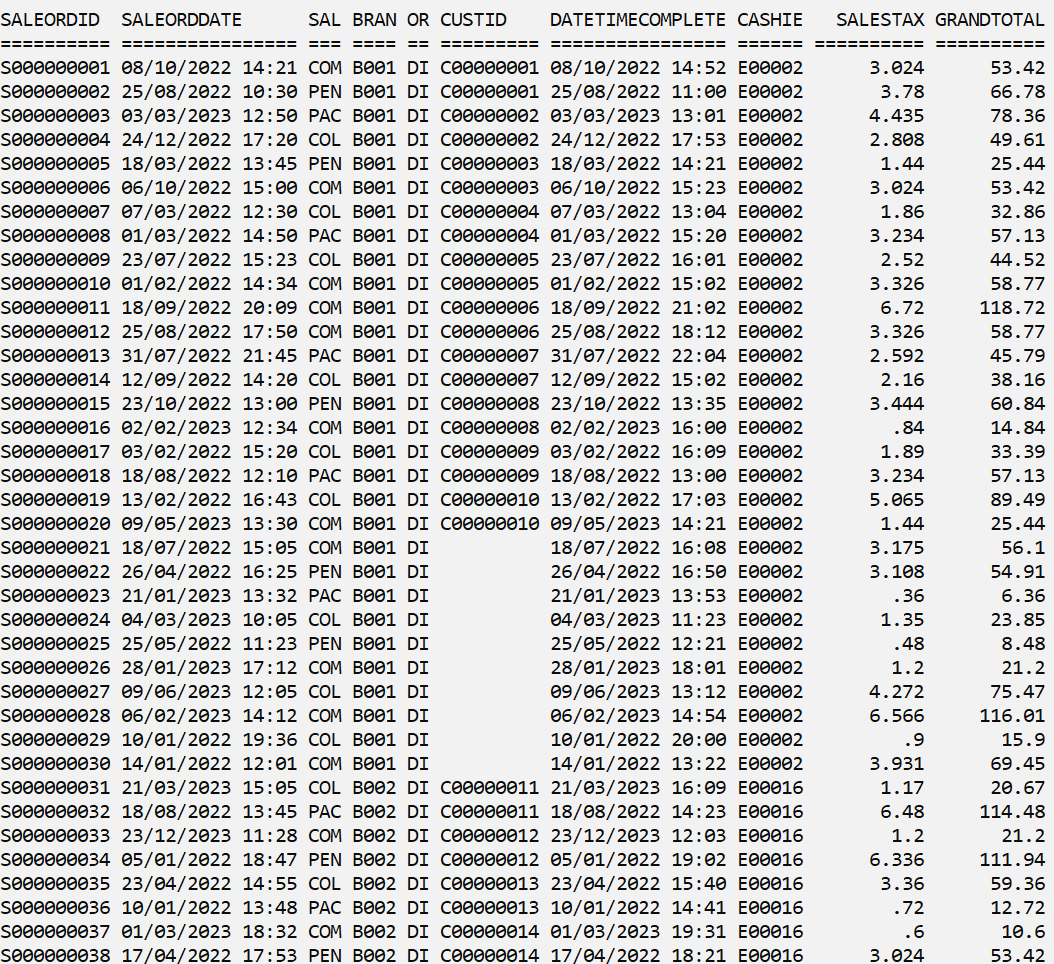
UpdateSaleOrder(p\_SaleOrdID => 'S000000300');

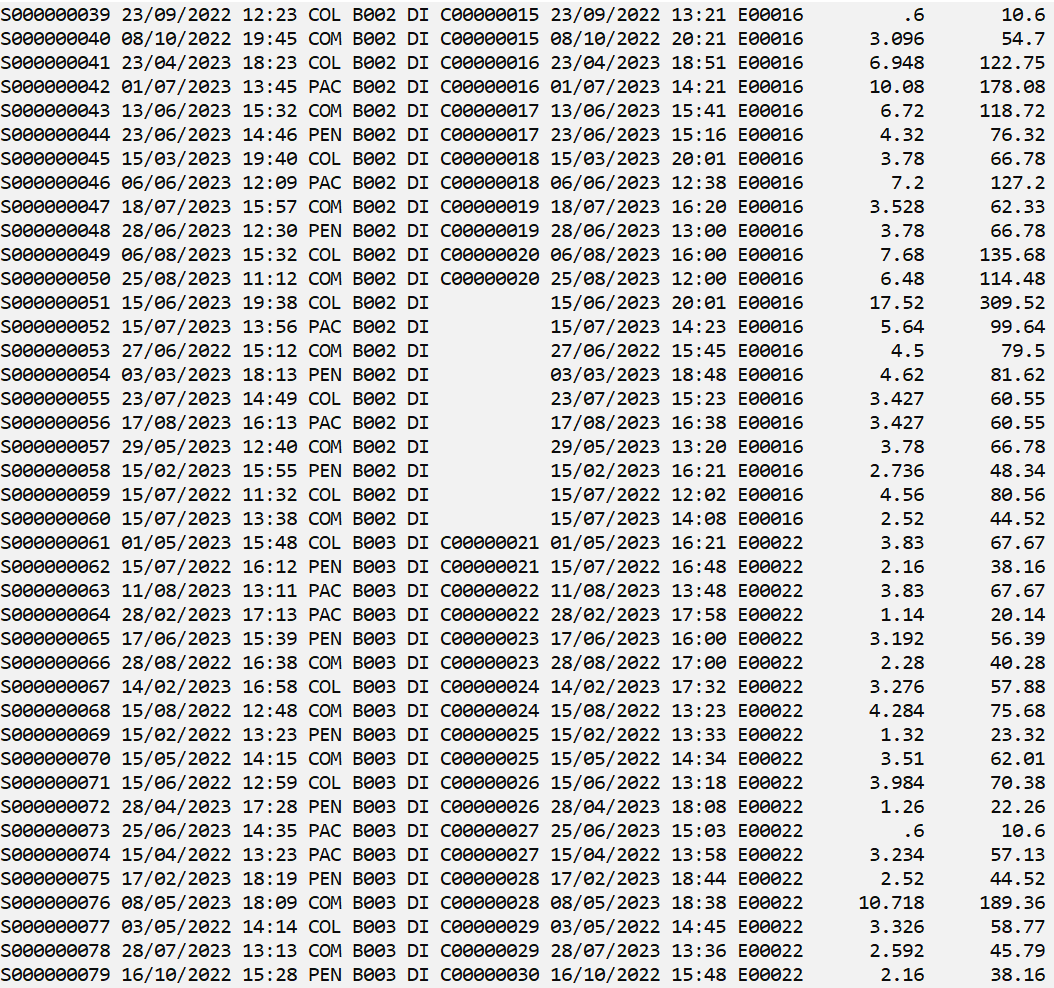
END;

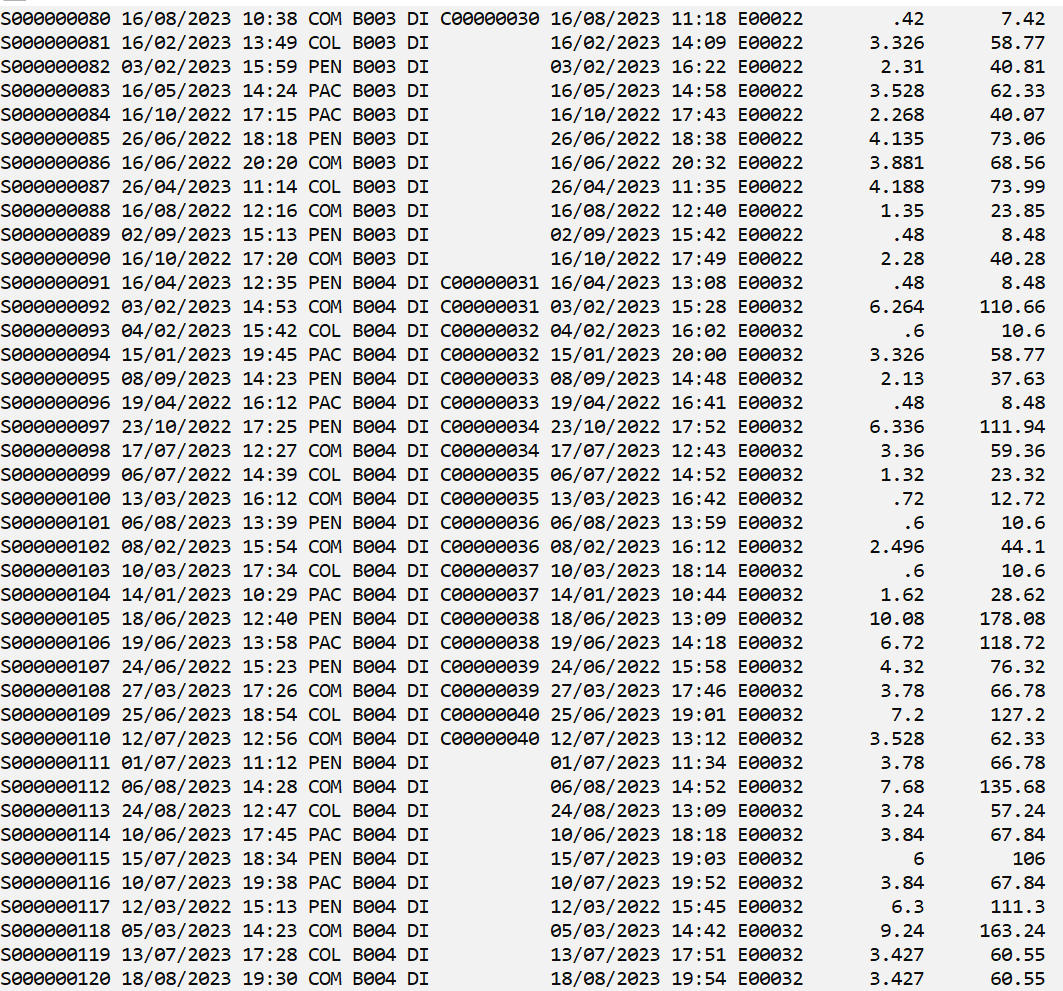
/

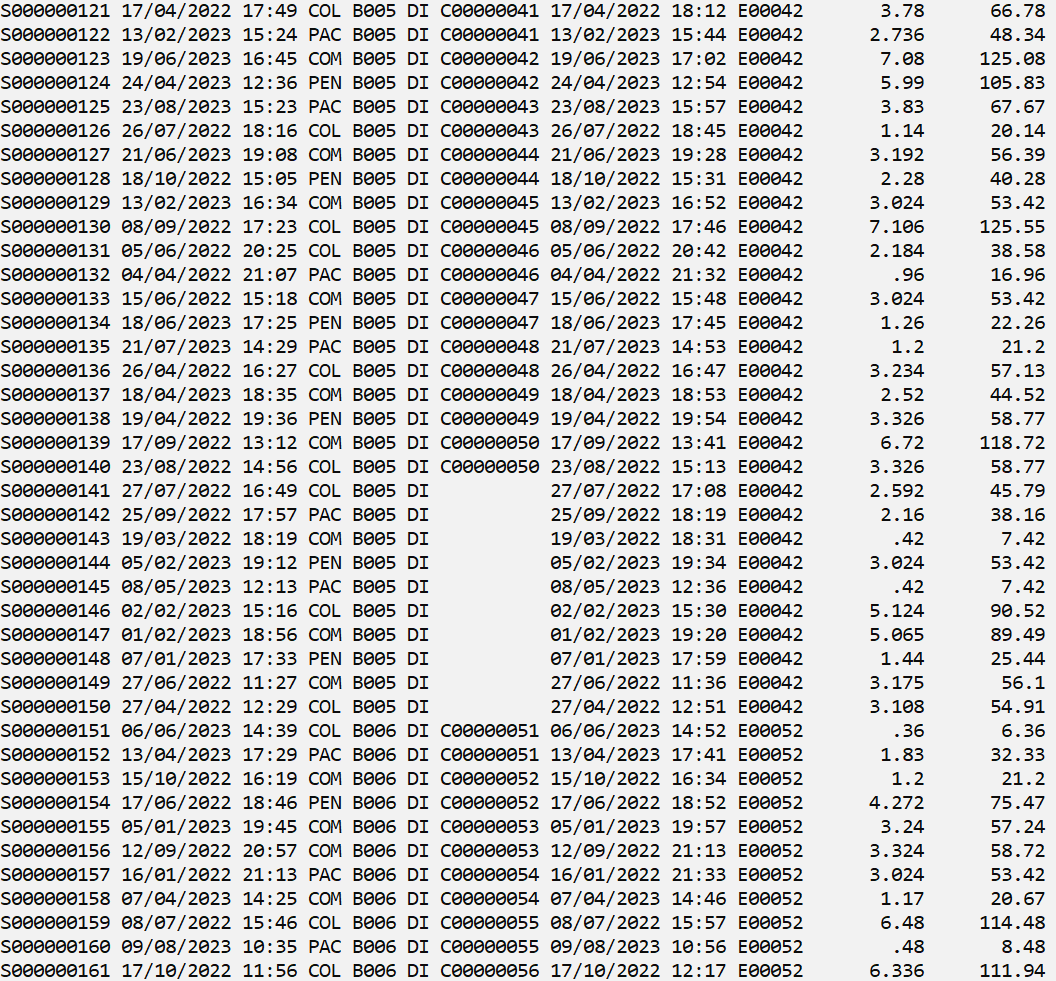
select \* from saleorder;

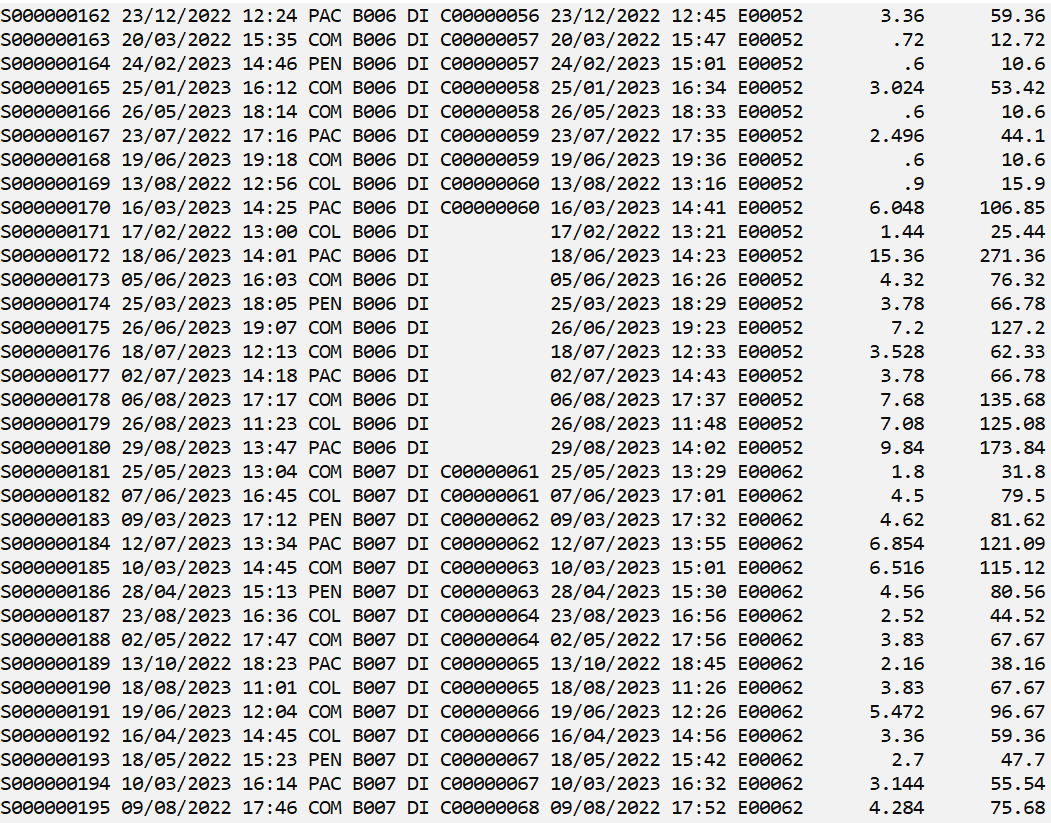
**Sample Output:**

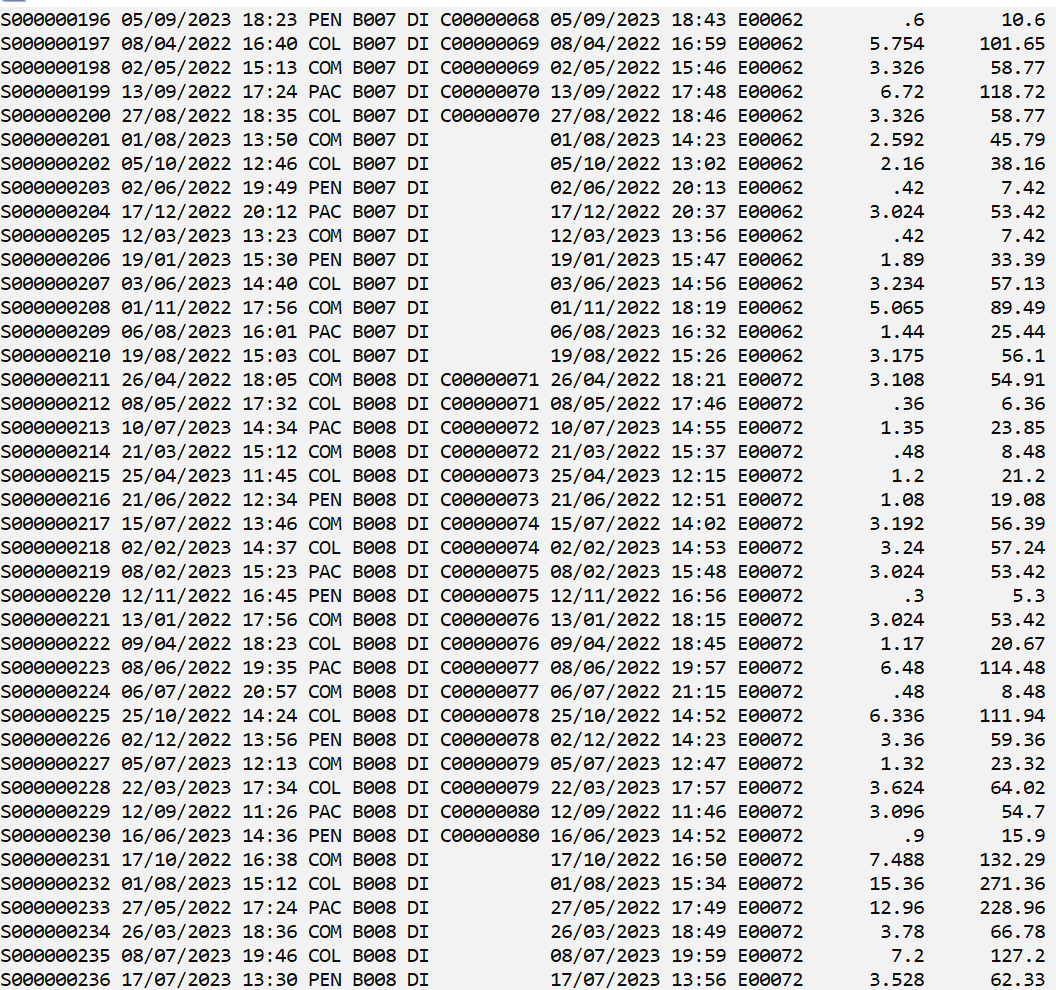


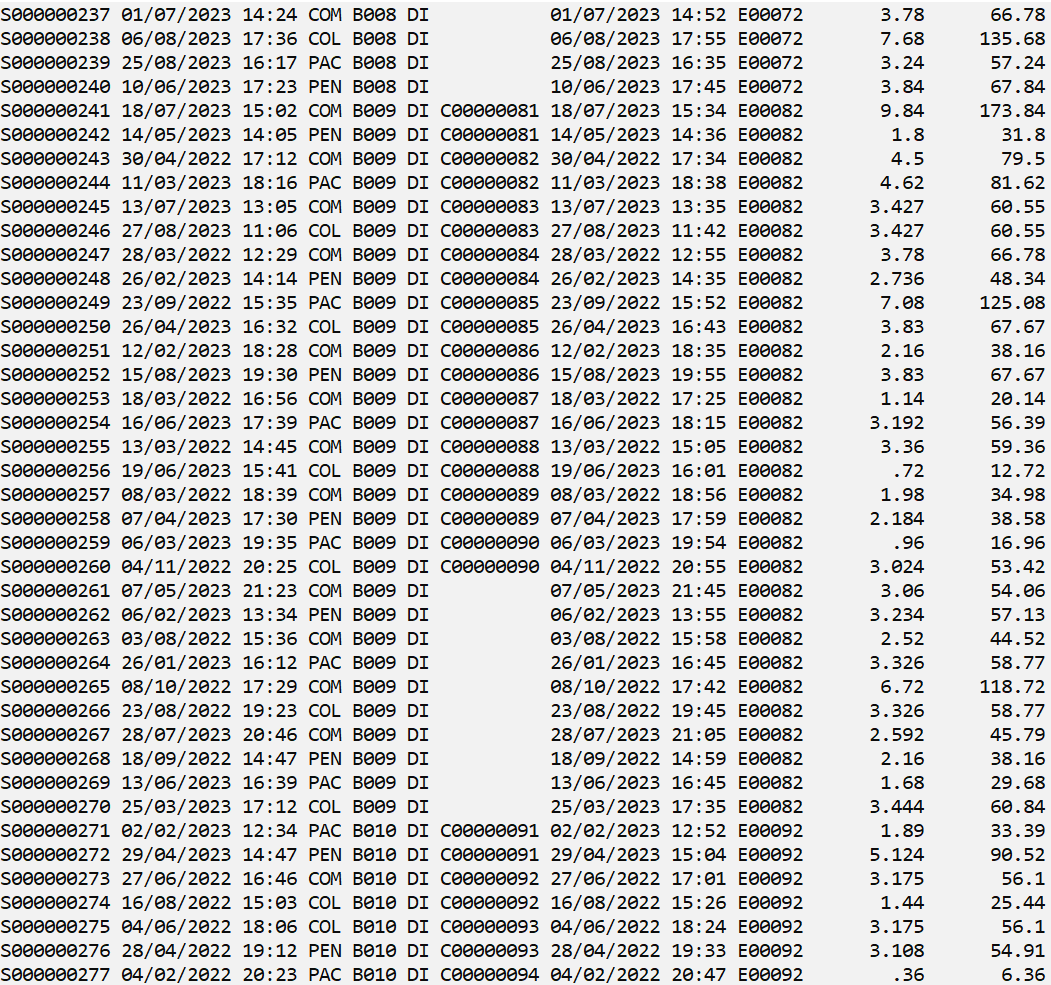


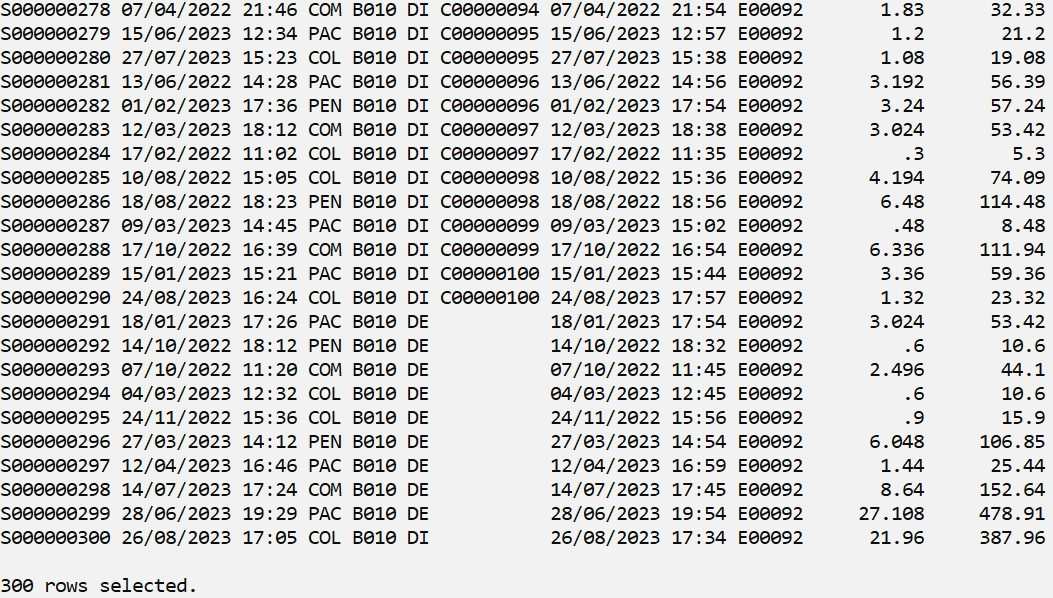












**4.4.6 Procedure 2: To update the order channel status for three different possibilities either DI which is Dine In or DE which is delivery or TA stands for take away. By determining by the customer id and the table no.**

**Purpose: The purpose of this procedure is to update the order channel in the sale order table. Because as users sit in our restaurant and dine in the status should be ‘DI’ as if the user didn’t dine in and also did not have a customer id then the status for order channel should be ‘TA’ which indicates take away. And if the user did not dine in but have a customer id it indicates ‘DE’ indicates delivery. With this the system can automatically indicate the status of the order channel by default and doesn’t need the user to manually edit the status.**

**Procedure code:**

CL SCR

CREATE OR REPLACE PROCEDURE pro\_update\_orderchl(

p\_SaleOrdID IN CHAR,

p\_TableNo NUMBER DEFAULT NULL

) IS

v\_custID char(9);

v\_orderChannel VARCHAR2(2) := 'TA';

BEGIN

SELECT CustID

INTO v\_custID

FROM SaleOrder

WHERE SaleOrdID = p\_SaleOrdID;

-- Adjust the order channel based on conditions (DI, DE, TA)

IF p\_TableNo IS NOT NULL THEN

v\_orderChannel := 'DI';

ELSE

IF v\_custID IS NULL THEN

v\_orderChannel := 'DE';

END IF;

END IF;

-- Update SaleOrder with the determined order channel

UPDATE SaleOrder

SET OrderChannel = v\_orderChannel

WHERE SaleOrdID = p\_SaleOrdID;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END pro\_update\_orderchl;

/

BEGIN

pro\_update\_orderchl('S000000291', NULL);

pro\_update\_orderchl('S000000292', NULL);

pro\_update\_orderchl('S000000293', NULL);

pro\_update\_orderchl('S000000294', NULL);

pro\_update\_orderchl('S000000295', NULL);

pro\_update\_orderchl('S000000296', NULL);

pro\_update\_orderchl('S000000297', NULL);

pro\_update\_orderchl('S000000298', NULL);

pro\_update\_orderchl('S000000299', NULL);

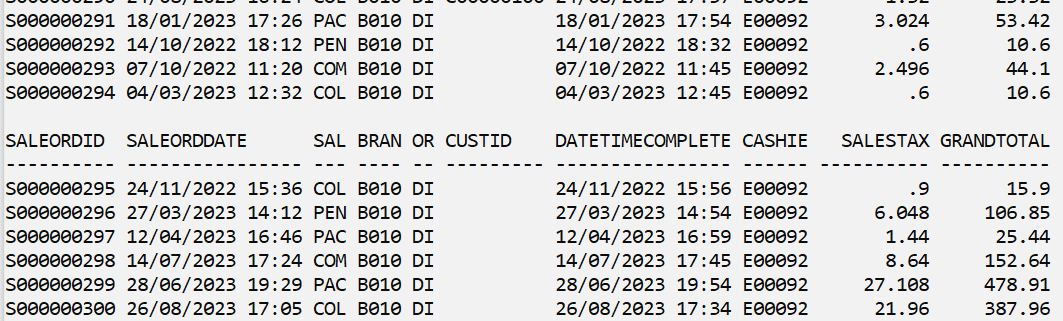
END;

/

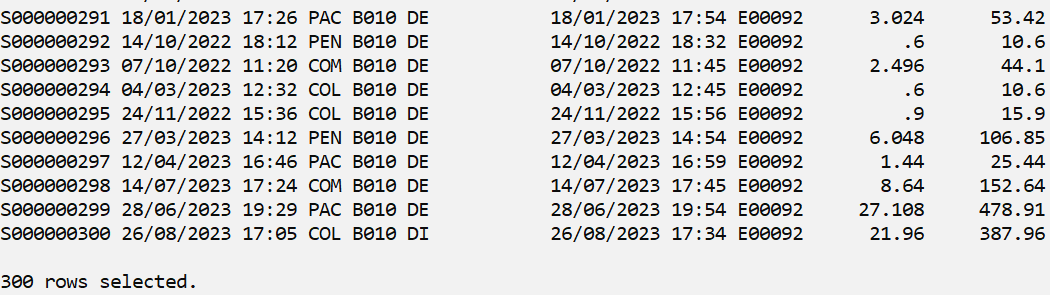
SELECT \* FROM SALEORDER;

**Sample Output:**

Before updating the order channel in sale order

****

After updating the order channel in sale order using procedure to update based on the requirement if fulfilled



**4.4.7 Report 1: Display the longest working employee, just joined employee and employees that worked more than 5 years in the fast food restaurant.**

**Purpose: The purpose of this report is to generate the longest working employee as to know the loyalty of the employee and also the employee that has just joined the company. Apart from that we also want to know the employees that have worked more than 5 years to gather the data for future improvements.**

**Report code:**

CL SCR

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE prc\_employee\_report AS

v\_EmployeeID CHAR(6);

v\_Name VARCHAR2(30);

v\_DateHired DATE;

v\_WorkingM10 DATE;

v\_WorkingM5 DATE;

-- store the current date for outer cursor

v\_CurrDate DATE;

v\_MinDateHired DATE;

--find the the dates of min and max

CURSOR m\_dates IS

SELECT MIN(DateHired) AS MinDate

FROM Employee

UNION

SELECT MAX(DateHired) AS MaxDate

FROM Employee;

CURSOR c\_employee(e\_date DATE) IS

SELECT EmployeeID, Name, DateHired

FROM Employee

WHERE DateHired = e\_date;

CURSOR c\_employee\_5years IS

SELECT EmployeeID, Name, DateHired

FROM Employee

WHERE DateHired <= ADD\_MONTHS(SYSDATE, - 60);

--12 \* 5 = 60 MONTHS, need two arguments for add\_months

BEGIN

SELECT MIN(DateHired) INTO v\_MinDateHired FROM Employee;

DBMS\_OUTPUT.PUT\_LINE(' EMPLOYEE REPORT ');

OPEN m\_dates;

LOOP

FETCH m\_dates INTO v\_CurrDate;

EXIT WHEN m\_dates%NOTFOUND;

-- inner cursor

OPEN c\_employee(v\_CurrDate);

FETCH c\_employee INTO v\_EmployeeID, v\_Name, v\_DateHired;

IF v\_DateHired = v\_MinDateHired THEN

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Longest Working Employee:');

DBMS\_OUTPUT.PUT\_LINE('EmployeeID: ' || v\_EmployeeID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_Name);

DBMS\_OUTPUT.PUT\_LINE('DateHired: ' || TO\_CHAR(v\_DateHired, 'DD/MM/YYYY'));

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');

ELSE

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Recently Joined Employee:');

DBMS\_OUTPUT.PUT\_LINE('EmployeeID: ' || v\_EmployeeID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_Name);

DBMS\_OUTPUT.PUT\_LINE('DateHired: ' || TO\_CHAR(v\_DateHired, 'DD/MM/YYYY'));

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');

OPEN c\_employee\_5years;

LOOP

FETCH c\_employee\_5years INTO v\_EmployeeID, v\_Name, v\_DateHired;

EXIT WHEN c\_employee\_5years%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Employees had work more than 5 years:');

DBMS\_OUTPUT.PUT\_LINE('EmployeeID: ' || v\_EmployeeID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_Name);

DBMS\_OUTPUT.PUT\_LINE('DateHired: ' || TO\_CHAR(v\_DateHired, 'DD/MM/YYYY'));

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');

END LOOP;

CLOSE c\_employee\_5years;

END IF;

CLOSE c\_employee;

END LOOP;

CLOSE m\_dates;

EXCEPTION

WHEN OTHERS THEN

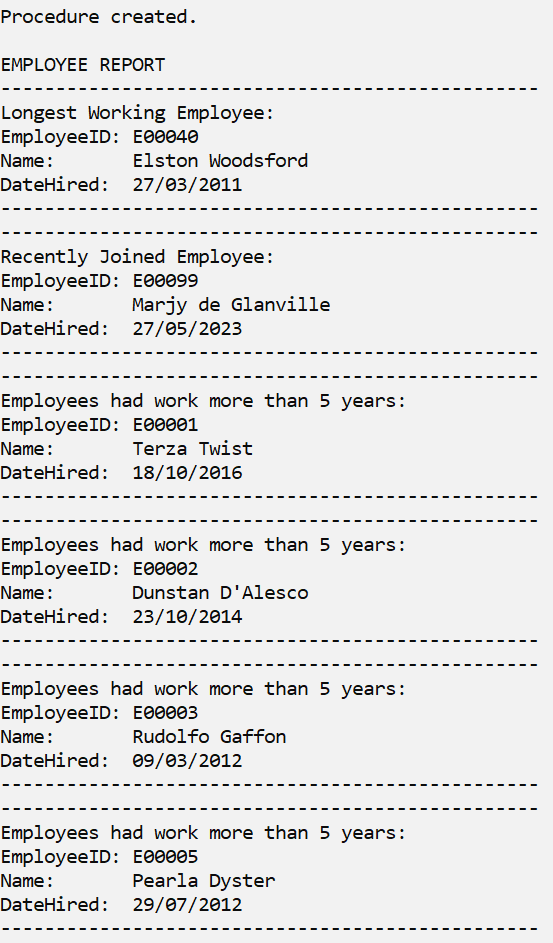
DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

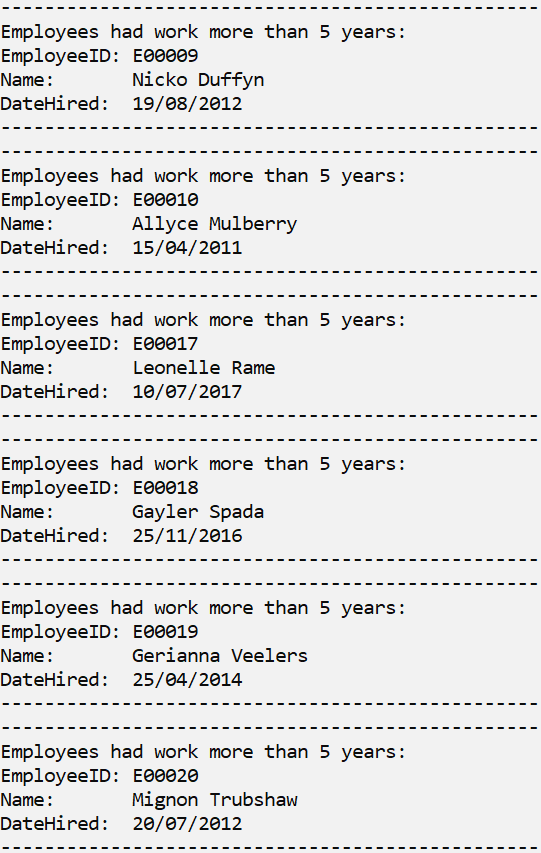
END prc\_employee\_report;

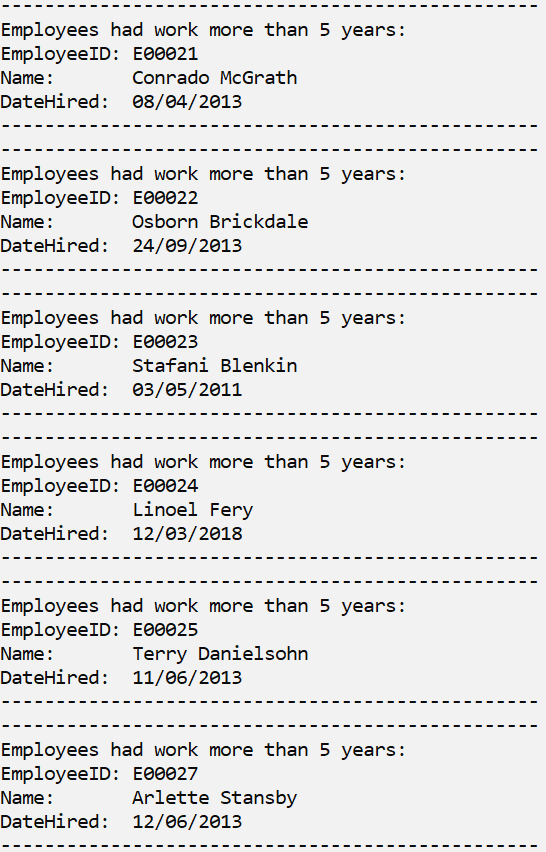
/

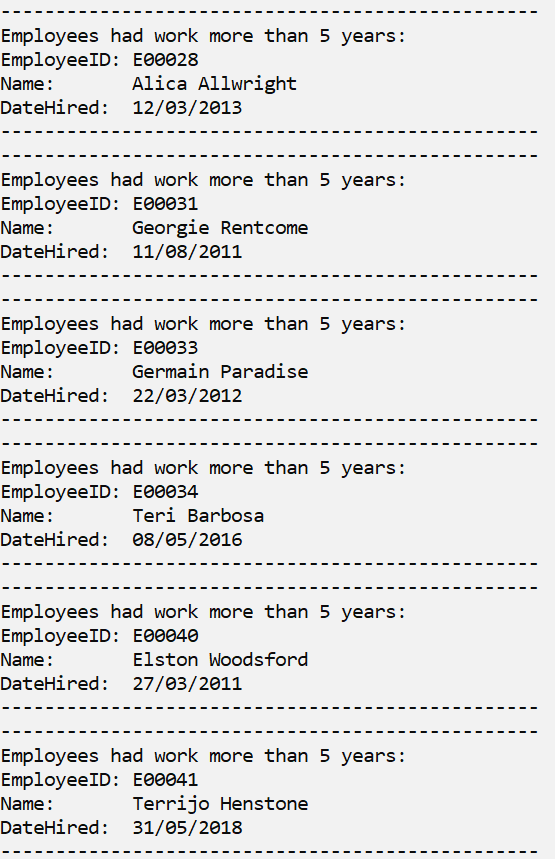
EXEC prc\_employee\_report;

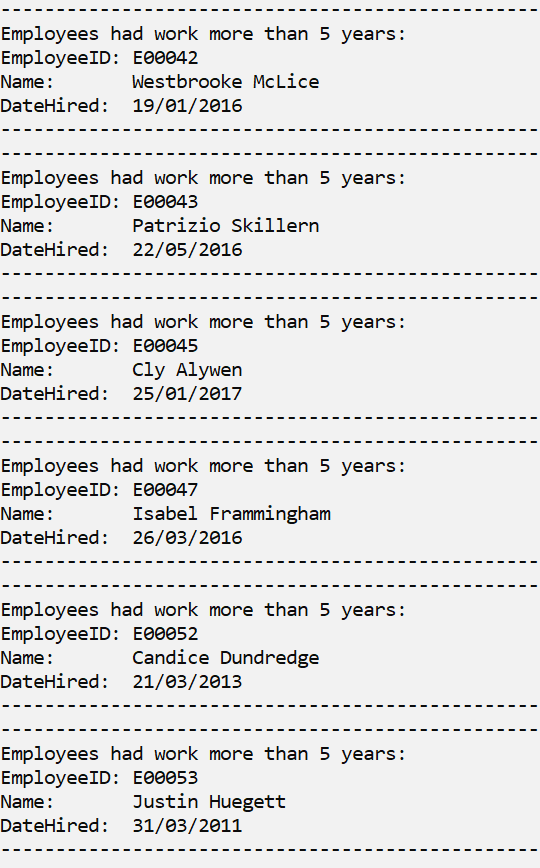
**Sample Output:**

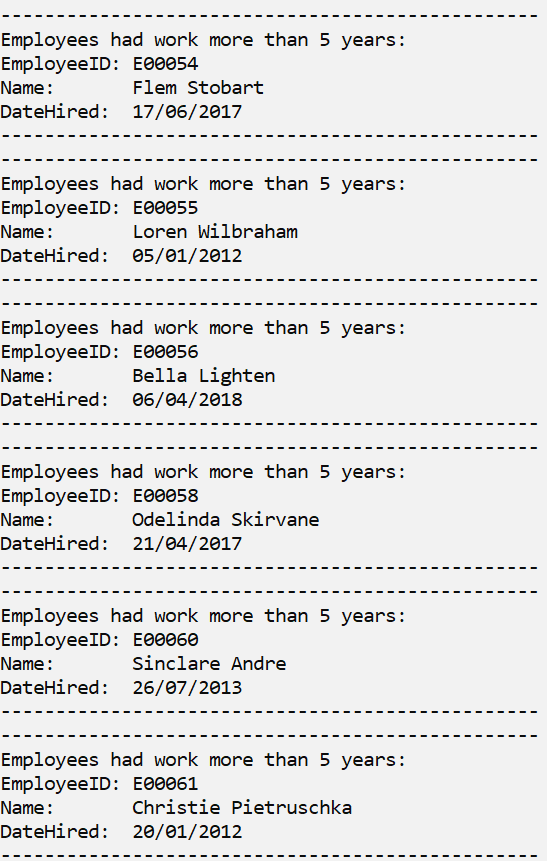


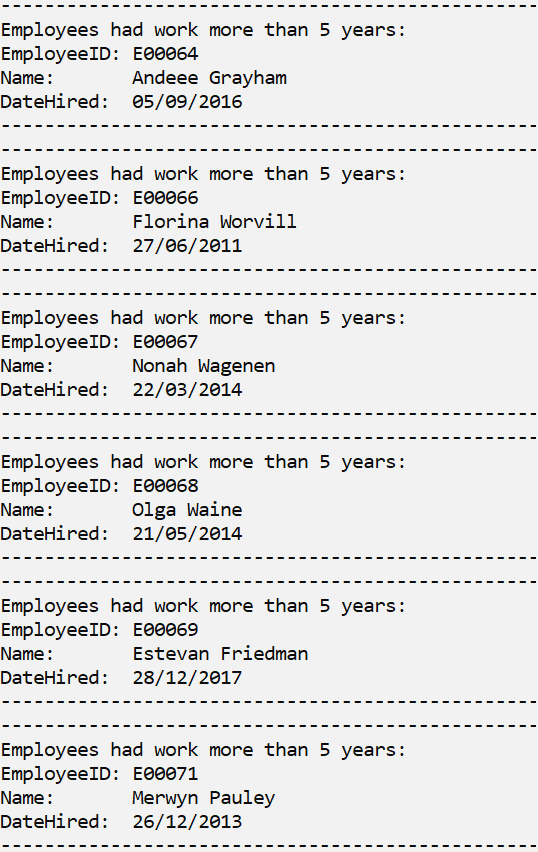


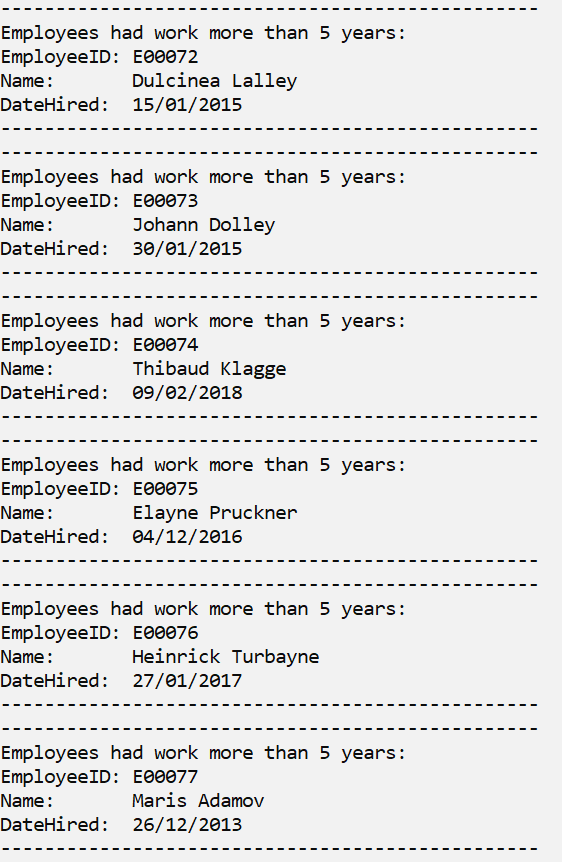


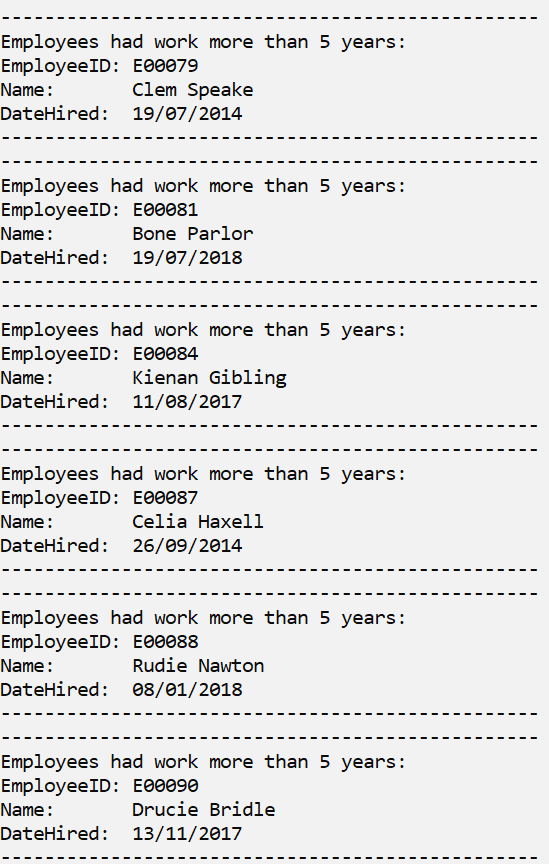


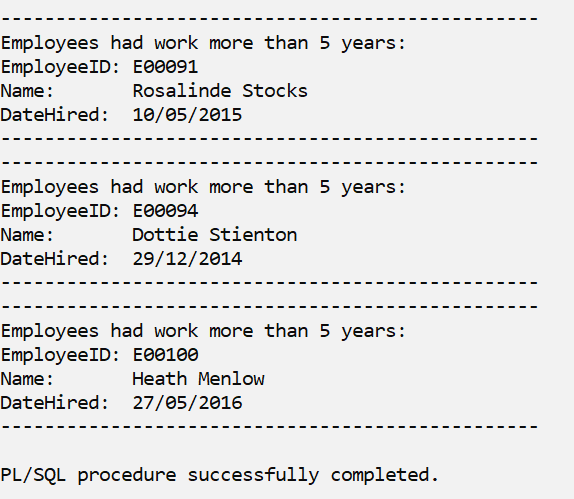












**4.4.8 Report 2: Displaying the revenue that the branches have generated in the 2 years.**

**Purpose: The purpose of this report is to generate revenue for the company, so that we can estimate the revenue we have gotten throughout the years and to make better informed decisions on how to improve the quality of our products, services and loyalty for the restaurant. As because after tax we can also find out the tax we had given out throughout these years.**

**Report Code:**

CL SCR

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE DisplayAnnualRevenue IS

v\_year NUMBER(4);

v\_annualRevenue NUMBER(8,2);

v\_totalRevenue NUMBER(8,2) := 0;

CURSOR c\_revenue IS

SELECT EXTRACT(YEAR FROM SaleOrdDate) AS OrderYear,

SUM(GrandTotal - SalesTax) AS Revenue

FROM SaleOrder

GROUP BY EXTRACT(YEAR FROM SaleOrdDate)

ORDER BY OrderYear;

BEGIN

OPEN c\_revenue;

DBMS\_OUTPUT.PUT\_LINE('Revenue By Year After Sales Tax:');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

LOOP

FETCH c\_revenue INTO v\_year, v\_annualRevenue;

EXIT WHEN c\_revenue%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Year ' || v\_year || ': ' || TO\_CHAR(v\_annualRevenue));

v\_totalRevenue := v\_totalRevenue + v\_annualRevenue;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Total Revenue: ' || TO\_CHAR(v\_totalRevenue));

CLOSE c\_revenue;

EXCEPTION

WHEN OTHERS THEN

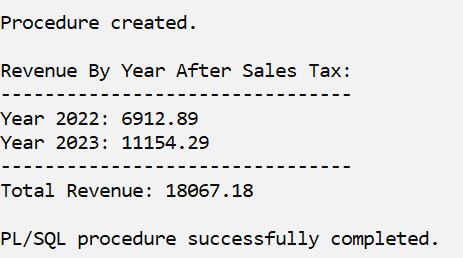
DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

EXEC DisplayAnnualRevenue;

**Sample Output:**



**Chapter 5 Extra Effort Highlights**

**5.1 (Ong Weng Kai)**

**5.1.1 Views**

CREATE VIEW CustomerSummary AS

SELECT

Branch.BranchID,

SUM(CASE WHEN Customer.Gender = 'M' THEN 1 ELSE 0 END) AS Total\_Male\_Cust,

SUM(CASE WHEN Customer.Gender = 'F' THEN 1 ELSE 0 END) AS Total\_Female\_Cust,

ROUND(AVG(CustomerFeedback.FoodRating), 2) AS Food\_Rating,

ROUND(AVG(CustomerFeedback.EnvironmentRating), 2) AS Environment\_Rating,

ROUND(AVG(CustomerFeedback.ServiceRating), 2) AS Service\_Rating,

ROUND(AVG(CustomerFeedback.PriceRating), 2) AS Price\_Rating

FROM

Branch

JOIN CustomerFeedback ON Branch.BranchID = CustomerFeedback.BranchID JOIN Customer ON CustomerFeedback.CustID = Customer.CustID

GROUP BY

Branch.BranchID;

SET LINESIZE 120

SELECT \* FROM CustomerSummary ORDER BY BranchID ASC;

DROP VIEW CustomerSummary;

**View is created for Query 1**

**5.1.4 User Defined Exceptions**

print out error message when exception occur

**Report 1 Exeception**

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END EmployeeReport;

**Report 2 Execption**

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END EmployeeReport;

**5.1.3 Sequences**

CREATE SEQUENCE SEQ\_BRANCH\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

CREATE SEQUENCE SEQ\_CUSTOMERFEEDBACK\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

CREATE SEQUENCE SEQ\_ING\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

**5.1.5 Column Formatting**

to make the code more understandable

**Procedure 2:**

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE(RPAD('BranchID', 10) || RPAD('FeedbackID', 15) || RPAD('Rating', 10) || 'Review');

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------');

**Trigger 1**

DBMS\_OUTPUT.PUT\_LINE('================================================');

DBMS\_OUTPUT.PUT\_LINE('Alert! Bad feedback received with FeedbackID: ' ||

**Trigger 2**

DBMS\_OUTPUT.PUT\_LINE('============================================================= ');

DBMS\_OUTPUT.PUT\_LINE('ALERT: The new menu item "' || :NEW.MenuDesc || '" (MenuID: ' || :NEW.MenuID || ') does not have an associated promotional discount. Consider adding one.');

**Report1**

DBMS\_OUTPUT.PUT\_LINE('Quarter Start: ' || TO\_CHAR(QuarterStart, 'DD/MM/YYYY'));

DBMS\_OUTPUT.PUT\_LINE('Food Rating : ' || AvgFoodRating);

DBMS\_OUTPUT.PUT\_LINE('Environment Rating: ' || AvgEnvironmentRating);

DBMS\_OUTPUT.PUT\_LINE('Service Rating : ' || AvgServiceRating);

DBMS\_OUTPUT.PUT\_LINE('Price Rating : ' || AvgPriceRating);

DBMS\_OUTPUT.PUT\_LINE('Total Customers : ' || TotalCustomers);

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------');

**Report 2**

DBMS\_OUTPUT.PUT\_LINE('Count of Male and Female Employees:');

DBMS\_OUTPUT.PUT\_LINE('----------------------------------');

OPEN gender\_cursor;

LOOP

FETCH gender\_cursor INTO v\_genderDesc, v\_count;

EXIT WHEN gender\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Gender: ' || v\_genderDesc || ' Count: ' || v\_count);

END LOOP;

CLOSE gender\_cursor;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Count of Employees for Each JobType:');

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

OPEN jobType\_cursor;

**5.1.6 Accept & Prompt**

To get input from the user so make the code to be more flexible, as they can show to get whatever information they needs instead of showing everything.

**Query 2**

ACCEPT IngredientCode CHAR PROMPT 'Please enter Ingredient Code > '

**5.2 (Ryan Kho Yuen Thian)**

**5.2.1 Views**

**Purpose: To get certain column(s) from the table and it is convenient to be used if I have to use it several times in the future**

CREATE OR REPLACE VIEW SetMeals AS

SELECT DISTINCT SetID

FROM MenuItem;

**Can be found in *Ryan - View1.txt.* This view is used for the 2nd query**

**5.2.2 Indexes**

**Purpose: Can help to speed up data retrieval for my query and reports**

DROP INDEX menu\_idx;

CREATE INDEX menu\_idx ON Menu(MenuItemCat, MenuID);

DROP INDEX saleOrd\_branchID\_idx;

CREATE INDEX saleOrd\_branchID\_idx ON SaleOrder(BranchID);

DROP INDEX purchOrder\_idx;

CREATE INDEX purchOrder\_idx ON PurchaseOrder(Status);

**Can be found in *Ryan - 3 indexes.txt*.**

**5.2.3 Sequences**

**Purpose: Convenient for inserting records into database since I do not have to manually keep track of the latest value of primary key value**

CREATE SEQUENCE SEQ\_CUST\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

CREATE SEQUENCE SEQ\_ING\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

CREATE SEQUENCE SEQ\_SET\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

CREATE SEQUENCE SEQ\_MENU\_ITEM\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

**5.2.4 Raise Application Error & Using System Defined Exception**

**Purpose: Used to point out the errors that may occur to the user when he/she is trying to process a delivered purchase order**

**i) Procedure 2’s Raise Application Error**

ELSIF v\_status = 'AR' THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Error: Purchase Order Status is already AR (ARRIVED)');

ELSIF v\_status = 'PA' THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error: Purchase Order Status is only PA (Pending Assignment to Supplier)');

ELSE

RAISE\_APPLICATION\_ERROR(-20003, 'Error: Invalid Purchase Order');

**ii) Procedure 2’s System Defined Exception**

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No record found! You may have entered wrong value(s)!');

END;

**5.2.5 Column Formatting**

**Purpose: To improve the appearance of 2 of my queries so that it is easily understandable**

**i) In Query 1**

TTITLE CENTER "The Fast Food Restaurant" SKIP 2 -

LEFT "Mkt Department" -

RIGHT "Branch Average Sales Revenues (Highest to Lowest) by Day of Week" SKIP 1 -

LEFT "=====================================================" -

"=================================================="

BTITLE LEFT "=====================================================" -

"==================================================" -

SKIP 1 -

RIGHT "Page " FORMAT 999 SQL.PNO

COLUMN BranchNo Heading "Branch No" FORMAT A9

COLUMN BranchName Heading "Branch Name"

COLUMN DayOfWeek HEADING "Day"

COLUMN BrnDowAvg HEADING "Average Sales (RM)" FORMAT 999,999.99

BREAK ON BranchNo SKIP 1 NODUPLICATES -

ON BranchName SKIP 1 NODUPLICATES -

**ii) In Query 2**

TTITLE CENTER "The Fast Food Restaurant" SKIP 2 -

CENTER "Ala Carte Products Currently on Promotion" SKIP 2 -

LEFT "LEGEND: CK = Chicken, BG = Burger, BV = Beverage, SD = Side Dish, ST = SET" SKIP 1 -

LEFT "==============================================================================================" -

"==========================================================================" -

BTITLE LEFT "==============================================================================================" -

"==========================================================================" -

CENTER "(Page " FORMAT 999 SQL.PNO ")"

COLUMN MenuItemCat HEADING "Category" FORMAT A8

COLUMN MenuID HEADING "Product ID" FORMAT A10

COLUMN MenuDesc HEADING "Product Name"

COLUMN BasePrice HEADING "Normal Price" FORMAT 999.99

COLUMN DiscountType HEADING "Discount Type"

COLUMN DiscountedPrice HEADING "Discounted Price" FORMAT 999.99

COLUMN DiscountPercent HEADING "Discount %" FORMAT 999.99

COLUMN NumberToBuy HEADING "No. to Buy (Free 1)" FORMAT 99

COLUMN MinOrderAmt HEADING "Min. Order Amt" FORMAT 999.99

COLUMN FromDate HEADING "Start Date"

COLUMN ToDate HEADING "End Date"

BREAK ON MenuItemCat SKIP 1 NODUPLICATES -

**5.2.6 Accept & Prompt**

**Purpose: To get input from user so that my query, report and procedures are more flexible and able to meet the user’s varying needs**

**i) Query 1**

PROMPT Please enter the following details

ACCEPT from\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter FROM Date > '

ACCEPT thru\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter TO Date > '

**ii) Report 1**

PROMPT Generate a Listing of Branches with Below Average Sales Revenues for a specified period

PROMPT -----------------------------------------------------------------------------------------

ACCEPT from\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter FROM Date: '

ACCEPT thru\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Enter TO Date : '

**iii) Procedure 1**

PROMPT

PROMPT =====================================================================

PROMPT Update Supplier, Qty and Price in Purchase Order

PROMPT =====================================================================

-- Prompt user to enter Key fields in order to retrieve Purchase Order

PROMPT Enter the Key Fields which identify the Purchase Order to be updated.

PROMPT

ACCEPT branch\_id PROMPT 'Branch ID > '

ACCEPT supplier\_id PROMPT 'Supplier ID > '

ACCEPT purch\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Purchase Date (DD/MM/YYYY) > '

ACCEPT ingredient\_id PROMPT 'Ingredient ID > '

SELECT \*

FROM PurchaseOrder

WHERE PurchaseOrder.BranchID = UPPER('&branch\_id') AND

PurchaseOrder.SupplierID = UPPER('&supplier\_id') AND

TO\_DATE(PurchaseOrder.PurchaseDate, 'DD/MM/YYYY') = TO\_DATE('&purch\_date','DD/MM/YYYY') AND

PurchaseOrder.IngID = UPPER('&ingredient\_id');

PROMPT Please enter additional details

ACCEPT new\_supplier\_id PROMPT 'New Supplier ID > '

ACCEPT order\_qty NUMBER FORMAT 9999999.99 PROMPT 'Order Quantity > '

ACCEPT order\_price NUMBER FORMAT 999.99 PROMPT 'Order Price (RM) > '

**iv) Procedure 2**

PROMPT

PROMPT =========================================================================================================

PROMPT Process Purchase Order Delivery

PROMPT =========================================================================================================

PROMPT To process a purchase order delivery, please enter the following details to retrieve the purchase order:

ACCEPT branch\_id PROMPT 'Branch ID > '

ACCEPT supplier\_id PROMPT 'Supplier ID > '

ACCEPT purch\_date DATE FORMAT 'DD/MM/YYYY' PROMPT 'Purchase Date (DD/MM/YYYY) > '

ACCEPT ingredient\_id PROMPT 'Ingredient ID > '

EXEC process\_po\_delivery\_sp('&branch\_id', '&supplier\_id', '&purch\_date', '&ingredient\_id');

**5.3 (Sim Hong Li)**

**5.3.1 Views**

**Views 1: Popular Delivery Company(use in Query 1)**

**Purpose: The view is to define the 3 popular delivery companies, because we total have 10 delivery companies, some of them not so famous in fast food delivery like ABX Express, ABC delivery. Limiting the companies will help restaurants focus on developing strategy on popular delivery companies.**

CREATE OR REPLACE VIEW Popular\_Delivery\_company\_VIEW AS

SELECT D.DeliveryCompanyID, D.DeliveryCompanyName, DI.SaleOrdID

FROM DeliveryCompany D,DeliveryOrderInfo DI

WHERE DI.DeliveryCompanyID = D.DeliveryCompanyID AND

DeliveryCompanyName IN ('GrabFood','Food Panda','KFC delivery');

**Views 2: Fresh ingredients(use in Query 2)**

**Purpose: Classify the type of ingredient**

drop VIEW fresh\_ingredient\_VIEW;

CREATE OR REPLACE VIEW fresh\_ingredient\_VIEW AS

SELECT \* FROM Ingredient

WHERE UPPER(IngType) IN ('MEA','VEG','BUN');

**5.3.2 Indexes**

**Purpose: to search the tables for the specific attributes faster**

CREATE INDEX idx\_MenuDiscToDate ON MenuDisc (ToDate);

**5.3.3 User Defined Exceptions**

**Purpose: Showing error message why the procedure dont work**

**Procedure 1**

INSERT\_WRONG\_SupplierID EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_SupplierID, -20001);

INSERT\_WRONG\_PurchaseDate EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_PurchaseDate, -20002);

INSERT\_WRONG\_IngreidientID EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_IngreidientID, -20003);

INSERT\_WRONG\_BranchID EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_BranchID, -20004);

**Procedure 2**

INSERT\_WRONG\_SalesOderId EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_WRONG\_SalesOderId, -20006);

**Trigger 1**

RAISE\_APPLICATION\_ERROR(-20900, 'Inventory is full for IngID: ' || :NEW.IngID || 'In Branch ' || :NEW.BranchID || ', Purchase Catalog Qty is ' || G\_TOTAL\_purQTY || ', larger than Max Inventory Qty ' || MAX\_QTY);

**Report 1**

RECORD\_FOUND EXCEPTION;

PRAGMA EXCEPTION\_INIT(RECORD\_FOUND , -20009);

**5.3.4 Sequences**

**Purpose: To make the id to a sequence for programmer no need to insert the id manually**

CREATE SEQUENCE SEQ\_SUPPLIER\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

**5.3.5 Formatting**

**Purpose: To make the output nicer for user to view for a easier understanding**

TTITLE CENTER 'ALL TIME POPULAR Primary Delivery Company In each Branch'-

RIGHT 'Page:' FORMAT 999 SQL.PNO SKIP 2

BREAK ON SupplierName ON ContactNo NODUP SKIP 1

TTITLE Left 'Least Purchase Ingredient Of Fresh Food Supplied ' 'Page:' FORMAT 999 SQL.PNO SKIP 2

**5.4 (Thong Cheng How)**

**5.4.1 Views**

**Purpose: To restrict the user to only see the tables cus, the user is required the state for the delivery (used in query 2)**

CREATE OR REPLACE VIEW del\_state AS

SELECT DISTINCT DelState FROM DeliveryOrderInfo Order by DelState;

SELECT \* FROM del\_state;

**5.4.2 Indexes**

**Purpose: to search the tables for the specific attributes faster**

CREATE INDEX idx\_saleorddate ON SaleOrder (SaleOrdDate);

CREATE INDEX idx\_orderItemStatus ON SaleOrderItem (OrderItemStatus);

CREATE INDEX idx\_datetimecompleted ON SaleOrder (DateTimeCompleted);

**5.4.3 Defined Exceptions**

**Purpose: to show the user error if the user has encountered an issue**

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

**5.4.4 User exception message**

**Purpose: Showing error message why the insert statement is not valid to be stored into the table (used in trigger 2)**

IF v\_yearsdiff < 17 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Employee should be at least 16 years old');

END IF;

**5.4.5 Sequence**

**Purpose: To make the id to a sequence for programmer no need to insert the id manually**

CREATE SEQUENCE SEQ\_SALEORDER\_ID

START WITH 1

INCREMENT BY 1

NOCACHE;

**5.4.6 accept and prompt**

**Purpose: To allow user to input data to search for the specific result. (used in query 1)**

ACCEPT sale\_order\_date\_int DATE FORMAT 'DD/MM/YYYY HH24:MI' PROMPT 'Please enter start date (DD/MM/YYYY HH:MI)->'

ACCEPT date\_completed\_int DATE FORMAT 'DD/MM/YYYY HH24:MI' PROMPT 'Please enter end date (DD/MM/YYYY HH:MI)->'

**5.4.7 formatting**

**Purpose: To make the output nicer for user to view for a easier understanding (used in query 1, procedure 2, report 1, report 2)**

TTITLE LEFT 'Branch Profitability Report' SKIP 2 -

LEFT 'Current Date: ' \_DATE SKIP 2

clear columns

clear breaks

clear computes

ttitle off

btitle off