IE2042 - Database Management Systems for Security

Assignment 01 : 2020 Regular Intake

Title : ***Supermarket***

Group Members: 04

IT Number	Name
IT19154640	WEERASIRI H.A.K.D.
IT19187488	Y.K.LUMINDU DILUMKA
IT19014432	YASHODHYA MADHUSHANI W.B.
IT19202228	SANDUNI KANCHANA A.M.K.

Contribution for the Project

IT Number	Contribution
IT19154640	• Identified and implemented the transaction to database. • Implemented countermeasure method of the database. • Partial contribution to ER Diagram. • Table creation and data insertion.
IT19187488	• Presenting the video regarding this Report. • Implemented access control privileges to database. • Partial contribution to ER Diagram. • Table creation and data insertion.
IT19014432	• Created Relation Schema. • explain the attacks and the countermeasures to the connected database is with a web application. • Partial contribution to ER Diagram. • Table creation and data insertion.
IT19202228	• Created Scenario. • Explained the recovery mechanisms regarding that type of database. • Partial contribution to ER Diagram. • Table creation and data insertion.

Table of Contents

DATABASE MANAGEMENT SYSTEM OF DUPERMAERKET

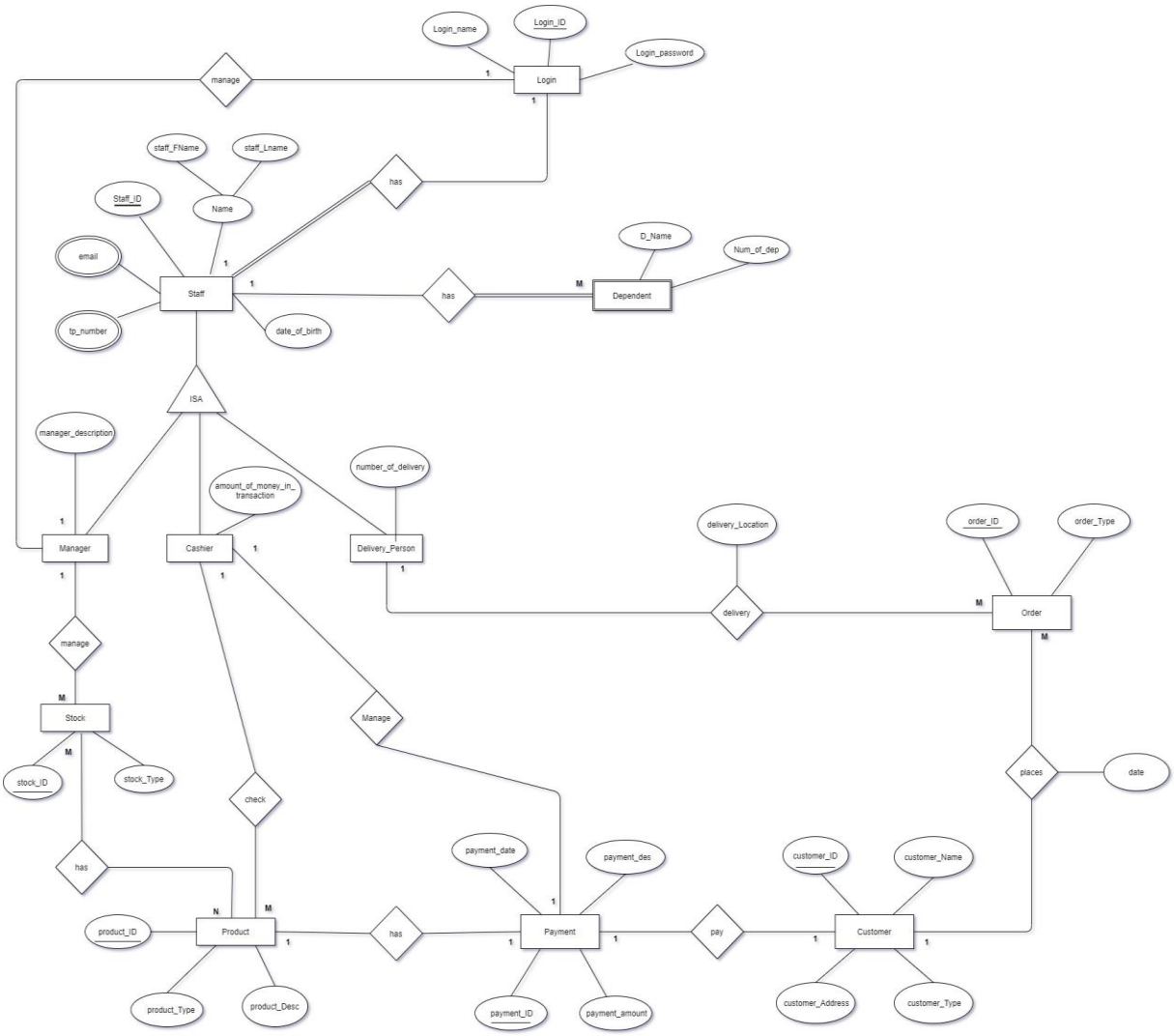
01. Main Scenario

Supermarket Database Management System is the concept behind this project. The store gets the sales products from many suppliers and the stock manager must enter the specifics of the shipment into the database until the merchandise are obtained and checked with the quantity and the price. Every category of item will include a stock ID and name. Each category is managed by different stock managers. Once the shopping has been completed, the customer will bring the shopping items to the cashier. Cashier will enter the product ID, product description, quantity and prices into the database and print the bill.

Afterwards the customer pays the bill by credit card, debit or cash. The cashier then enters the invoice information into the database. Each payment has a unique payment ID, and the dates and times for payment are documented in the database.

Data may only be checked and modified for security purposes by the manager. The manager is responsible for maintaining the database, and he is given all rights such as adding, modifying, deleting in the database.

02. ER Diagram



03. Schema Of The Database

Login

Login_ID	Login_name	Login_password

Staff

Staff _ID		Staff _Fname	Staff _Lname	Date-of-birth	Login_ID

Dependent

Staff _ID	D_Name	Num_Of_Dep

Manager

Staff_ID	manager_description

Cashier

Staff _ID	amount_of_money_in_transaction

Delivery Person

Staff_ID	number_of_delivery

Staff_ tp_number

Staff _ID	Contact_no

Stock

Stock_ID	Quantity	stock_Type	Staff _ID

Product

Product_ID	Product_type	Product_Desc	Staff_ID

Stock_Product

Stock_ID	Product_ID

Payment

Payment_ID	Payment_amount	Payment_date	Payment_des	Staff_ID

Customer

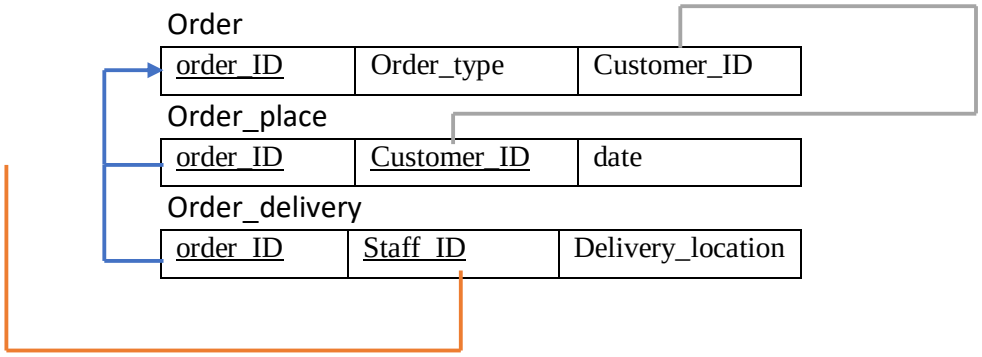
Customer_ID	Customer_name	Customer_address	Customer_type	Payment_ID

Customer_contact

Customer_ID	Customer_name

Customer_email

Customer_ID	Cust_mail


Order
order_ID
Order_type
Customer_ID
Order_place
order_ID
Customer_ID
date
Order_delivery
order_ID
Staff_ID
Delivery_location

04. Table creation queries

Stock table

```
SET DEFINE OFF;
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK001','S0011','common stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK002','S0010','common stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK003','S0012','preferred stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK004','S0010','preferred stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK005','S0011','common stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK006','S0012','preferred stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK007','S0011','common stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK008','S0010','common stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK009','S0010','common stock');
INSERT INTO Stock(stock_ID,staff_ID,stock_Type)VALUES('SK000','S0012','preferred stock');
```

Product table

```
SET DEFINE OFF;
INSERT INTO Product VALUES ('p0001','SK001','convenience','Toothpaste,cake');
INSERT INTO Product VALUES ('p0002','SK002','shopping','Milk Bottle,Books,bags');
INSERT INTO Product VALUES ('p0003','SK003','shopping','Water Bottles,Thorn work papers');
INSERT INTO Product VALUES ('p0005','SK005','shopiing','Bags','s0014');
INSERT INTO Product VALUES ('p0006','SK006','convenience','Gifts');
INSERT INTO Product VALUES ('p0007','SK007','convenience','cake mix');
INSERT INTO Product VALUES ('p0008','SK008','shopping','Foods');
INSERT INTO Product VALUES ('p0009','SK009','shopping','Toothpaste');
INSERT INTO Product VALUES ('p0010','SK000','convenience','Books,Bags');
```

Payment table

```
SET DEFINE OFF;
INSERT INTO Payment VALUES('P1','p0001','1000',to_date('12-MAR-2019','DD-MON-RR'),'Pay Pal');
INSERT INTO Payment VALUES('P2','p0002','560',to_date('12-OCT-2019','DD-MON-RR'),'On Delivery');
INSERT INTO Payment VALUES('P3','p0003','5000',to_date('22-APR-2019','DD-MON-RR'),'Credit Card');
INSERT INTO Payment VALUES('P4','p0004','460',to_date('14-JUN-2019','DD-MON-RR'),'Pay Pal');
INSERT INTO Payment VALUES('P5','p0005','1500',to_date('02-AUG-2019','DD-MON-RR'),'Credit Card');
INSERT INTO Payment VALUES('P6','p0006','1700',to_date('17-DEC-2019','DD-MON-RR'),'Pay Pal');
INSERT INTO Payment VALUES('P7','p0007','1560',to_date('26-MAR-2019','DD-MON-RR'),'On Delivery');
INSERT INTO Payment VALUES('P8','p0008','5200',to_date('16-FEB-2019','DD-MON-RR'),'Credit Card');
INSERT INTO Payment VALUES('P9','p0001','1460',to_date('07-MAY-2019','DD-MON-RR'),'Pay Pal');
INSERT INTO Payment VALUES('P0','p0003','1800',to_date('09-JUL-2019','DD-MON-RR'),'Credit Card');
```

05. All tables after inserting data

1.Login table

	LOGIN_ID	LOGIN_NAME	LOGIN_PASSWORD
1	L0124	Lahiru	ytLm123#w1
2	L0125	Eranga	k12#aqeftK
3	L0134	Kavindu	424Aqef#e1
4	L0135	Nuwanthi	Lq12dr5&fe
5	L0140	Nimesh	KL12w$qmn9
6	L0143	Jayani	ewdee#6gt
7	L0144	Dinithi	zw$def#6ht
8	L0145	Gethmi	awdef#6ht&
9	L0150	Chamani	123ef#ghg&
10	L0152	Amila	564$kmql&8
11	L0155	Vinsadi	qswemhfl&1
12	L0156	Sahan	1234qwvgr&
13	L0123	Jetumini	LwseM1264
14	L0157	Yasuri	degtL12%km
15	L0160	Apeksha	0973jhebrn
16	L0161	Hasitha	lkTa123J&m

2. Dependent table

	STAFF_ID	D_NAME	NUM_OF_DEP
1	S0010	Lumindu	DPT01
2	S0011	Saranaga	DPT07
3	S0012	null	NULL
4	S0013	Dakshina	DPT02
5	S0014	NULL	NULL
6	S0015	Sanduni	DPT04
7	S0016	NULL	NULL
8	S0017	Praveen	DPT10
9	S0018	Samalka	DPT14
10	S0019	NULL	NULL
11	S0020	Vidun	DPT09
12	S0024	Malith	DPT05
13	S0025	NULL	NULL
14	S0026	Saranaga	DPT07
15	S0027	NULL	NULL
16	S0028	Awantha	DPT16

3.Staff table

	LOGIN_ID	STAFF_ID	STAFF_FNAME	STAFF_LNAME	DATE_OF_BIRTH
1	L0124	S0010	Lahiru	Munasinghe	23-FEB-96
2	L0125	S0011	Eranga	Gimsara	20-DEC-97
3	L0134	S0012	Kavindu	Jayasinghe	12-SEP-98
4	L0135	S0013	Nuwanthi	Madushika	23-OCT-94
5	L0140	S0014	Nimesh	Maduwantha	10-MAR-95
6	L0143	S0015	Jayani	Dilmini	19-JUN-99
7	L0144	S0016	Dinithi	Mohotty	01-JAN-98
8	L0145	S0017	Gethmi	Kaveesha	09-FEB-95
9	L0150	S0018	Chamani	Jayanka	31-OCT-99
10	L0152	S0019	Amila	Rathnayaka	12-FEB-98
11	L0155	S0020	Vinsadi	Sinethma	26-JUL-94
12	L0156	S0024	sahan	Randima	25-NOV-95
13	L0123	S0025	Jetumini	Imanahari	15-DEC-95
14	L0157	S0026	Yasuri	Bandaara	25-DEC-95
15	L0160	S0027	Apeksha	Warnakulasuriya	15-APR-99
16	L0161	S0028	Hasitha	Suraweera	02-FEB-95

4.Cashier table

	STAFF_ID	AMOUNT_OF_MONEY_IN_TRANSACTION
1	S0013	1200
2	S0014	5000
3	S0015	4300
4	S0016	7600
5	S0017	9800

5. Manager table

	STAFF_ID	MANAGER_DESCRIPTION
1	S0010	(null)
2	S0011	(null)
3	S0012	(null)

6.Delivery person table

	STAFF_ID	NUMBER_OF_DELIVERY
1	S0018	12
2	S0019	8
3	S0020	3
4	S0024	5
5	S0025	6
6	S0026	10
7	S0027	4
8	S0028	6

7.staff_mobile table

	STAFF_ID	TP_NUMBER
1	S0010	0711789418
2	S0012	0763498820
3	S0011	0712823733
4	S0011	0412231874
5	S0014	0112468100
6	S0013	0711219145
7	S0019	0112233445
8	S0028	0702424345
9	S0013	0114566324
10	S0027	0753634267
11	S0027	0777864356
12	S0018	0713425643
13	S0019	0716746534
14	S0016	0715467544
15	S0016	0762646733
16	S0026	0754575356
17	S0028	0778546342
18	S0027	0752567434
19	S0018	0713534343
20	S0014	0723535464
21	S0015	0112465343

8.Staff email table

	STAFF_ID	EMAIL
1	S0014	kevindk@gmail.com
2	S0016	jayani432@gmail.com
3	S0012	sahan3000@gmail.com
4	S0015	hashanmadu32@gmail.com
5	S0011	sanduni1998@gmail.com
6	S0014	dinithi234@gmail.com
7	S0015	nuwanthi765@gmail.com
8	S0019	gethmi5432@gmail.com
9	S0028	tharidu432@gmail.com
10	S0018	keshu1234@gmail.com
11	S0021	nuwanthi765@gmail.com
12	S0028	summer.payne@outlook.com
13	S0018	rose.stephens@gmail.com
14	S0019	annabelle.dunn@outlook.com
15	S0018	tommy.bailey@gmail.com
16	S0016	jude.rivera@gmail.com
17	S0026	jude.rivera@outlook.com
18	S0024	elliot.brooks@gmail.com
19	S0027	albert.watson@gmail.com
20	S0018	mohammad.peterson@outlook.com
21	S0014	harper.spencer@gmail.com
22	S0016	louierichardson@yahoo.com

9.Stock table

	STOCK_ID	STAFF_ID	STOCK_TYPE
1	SK001	S0011	common stock
2	SK002	S0010	common stock
3	SK003	S0012	preferred stock
4	SK004	S0010	preferred stock
5	SK005	S0011	common stock
6	SK006	S0012	preferred stock
7	SK007	S0011	common stock
8	SK008	S0010	common stock
9	SK009	S0010	common stock
10	SK000	S0012	preferred stock

10.Product table

	PRODUCT_ID	STOCK_ID	PRODUCT_TYPE	PRODUCT_DESC
1	p0001	SK001	convenience	Toothpaste,cake
2	p0002	SK002	shopping	Milk Bottle,Books,bags
3	p0003	SK003	shopping	Water Bottles,Thorn work papers
4	p0006	SK006	convenience	Gifts
5	p0007	SK007	convenience	cake mix
6	p0008	SK008	shopping	Foods
7	p0009	SK009	shopping	Toothpaste
8	p0010	SK000	convenience	Books,Bags

11.Payment table

	PAYMENT_ID	PRODUCT_ID	PAYMENT_AMOUNT	PAYMENT_DATE	PAYMENT_DESC
1	P1	p0001	1000	12-MAR-19	Pay Pal
2	P2	p0002	560	12-OCT-19	On Delivery
3	P3	p0003	5000	22-APR-19	Credit Card
4	P6	p0006	1700	17-DEC-19	Pay Pal
5	P7	p0007	1560	26-MAR-19	On Delivery
6	P8	p0008	5200	16-FEB-19	Credit Card
7	P9	p0001	1460	07-MAY-19	Pay Pal
8	P0	p0003	1800	09-JUL-19	Credit Card

12.Customer table

	CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CUSTOMER_TYPE	PAYMENT_ID
1	CST00	Sahan	10315 Hickman Rd, Des Moines, IA	Platinum	P1
2	CST01	Janith	3324 N Oakland Ave, Milwaukee, WI	Signature	P2
3	CST02	Kasun	1613 Victoria St, Calcutta	Preimier	P3
4	CST03	Nimal	Via Dolorosa 69, Roma	Preimier	P4
5	CST04	Ravidu	310 Broadway St, Alexandria, MN	Platinum	P5
6	CST05	Eranga	660 Woodward Ave # 2290, Detroit, MI	Platinum	P6
7	CST06	Venura	1592 Silverado St, Bangalore, Kar	Signature	P7
8	CST07	Bhanuka	215 4Th Ave Se, Cedar Rapids, IA	Preimier	P8
9	CST08	Dinuka	6Th And Master St, Philadelphia, PA	Signature	P9
10	CST09	Bryan	Welschdoerfchen 1941, Chur, ZH	Platinum	P10

13.Cust_contact table

	CUSTOMER_ID	CUST_NUMB
1	CST00	777382648
2	CST01	774437599
3	CST01	714658976
4	CST02	714466217
5	CST03	715573884
6	CST04	751123259
7	CST05	777721344
8	CST05	772124375
9	CST06	713354784
10	CST07	712234585
11	CST08	771253547
12	CST09	763213658
13	CST09	762235475

14.Cust_email

	CUSTOMER_ID	CUST_MAIL
1	CST00	sahan123@gmail.com
2	CST00	sahan7@gmail.com
3	CST01	janih@gmail.com
4	CST01	janith99@gmail.com
5	CST02	kasun@gmail.com
6	CST02	kasun78@gmail.com
7	CST03	nimal@gmail.com
8	CST04	ravidu45@gmail.com
9	CST05	eranga@gmail.com
10	CST06	venura95@gmail.com
11	CST07	bhanuka99@gmail.com
12	CST08	dinuka21@gmail.com
13	CST09	bryan66@gmail.com

15.Order table

	ORDER_ID	ORDER_TYPE	CUSTOMER_ID	STAFF_ID
1	ORD00	priority	CST00	S0018
2	ORD01	priority	CST01	S0019
3	ORD02	non-priority	CST02	S0020
4	ORD04	non-priority	CST04	S0025
5	ORD05	non-priority	CST05	S0024
6	ORD06	priority	CST06	S0026
7	ORD07	priority	CST07	S0027
8	ORD08	priority	CST08	S0028
9	ORD09	non-priority	CST09	S0018

16.Order_place table

	CUSTOMER_ID	ORDER_ID	DATE_OF_PLACE
1	CST00	ORD00	03-MAR-20
2	CST01	ORD01	26-JUL-20
3	CST02	ORD02	06-FEB-20
4	CST03	ORD03	13-JAN-94
5	CST04	ORD04	01-FEB-20
6	CST05	ORD05	09-APR-20
7	CST06	ORD06	16-FEB-20
8	CST07	ORD07	14-JAN-20
9	CST08	ORD08	19-JAN-20
10	CST09	ORD09	28-FEB-20

17.Order_delivery table

	ORDER_ID	STAFF_ID	DELIVERY_LOCATION
1	ORD00	S0018	10315 Hickman Rd, Des Moines, IA
2	ORD01	S0020	3324 N Oakland Ave, Milwaukee, WI
3	ORD02	S0028	1613 Victoria St, Calcutta
4	ORD03	S0024	Via Dolorosa 69, Roma
5	ORD04	S0019	310 Broadway St, Alexandria, MN
6	ORD05	S0026	660 Woodward Ave # 2290, Detroit, MI
7	ORD06	S0028	1592 Silverado St, Bangalore, Kar
8	ORD07	S0028	215 4Th Ave Se, Cedar Rapids, IA
9	ORD08	S0027	6Th And Master St, Philadelphia, PA
10	ORD09	S0020	Welschdoerfchen 1941, Chur, ZH

06. Transactions of the database

A transaction is a logical working unit that contains one or more SQL statements. An Atomic Unit is a transaction. The effects of all SQL statements in the transaction can be either all committed (applied to the database) or all rolled back (undone from the database).

The transaction starts with the first executable SQL statement. The transaction ends when committed or rollback, with either a COMMIT declaration or a ROLLBACK declaration or with an implied DDL declaration.

Here are the some transactions with explaining its works.

- Updating Dependent table

```
SET TRANSACTION NAME 'Dependent_Update';
select * from dependent;

update dependent
SET d_name = 'Rasil'
WHERE staff_id = 'S0012';

COMMIT;

SAVEPOINT after_dep_update;
```

It shows Updating of Dependent Table in that sentence.

First of all, SET TRANSACTION NAME 'Dependent Update' says this statement starts a transaction and calls it 'Dependent Update'. Before that, it must recover all the data before the changes are made.

Worksheet | Query Builder

```
select * from dependent;
```

Script Output | Query Result

All Rows Fetched: 16 in 0.852 seconds

	STAFF_ID	D_NAME	NUM_OF_DEP
1	S0010	Lumindu	DPT01
2	S0011	Saranaga	DPT07
3	S0012	null	NULL
4	S0013	Dakshina	DPT02
5	S0014	NULL	NULL
6	S0015	Sanduni	DPT04
7	S0016	NULL	NULL
8	S0017	Praveen	DPT10
9	S0018	Samalka	DPT14
10	S0019	NULL	NULL
11	S0020	Vidun	DPT09
12	S0024	Malith	DPT05
13	S0025	NULL	NULL
14	S0026	Saranaga	DPT07
15	S0027	NULL	NULL
16	S0028	Awantha	DPT16

Highlighted with the blue color is shows the what is going to update. After that it Update the ID of **'S0012'** to dependent name as **'Rasil'.**

In that code, 'COMMIT' statement says that ends any existing transaction in the session.

Oracle had created undo information prior to that declaration. The undo information includes the old data values which are modified by the transaction's SQL statements.

But after the commit, the transaction's respective specific system change number (SCN) is allocated and reported in the table.

Here's after the code section commit.

	STAFF_ID	D_NAME	NUM_OF_DEP
1	S0010	Lumindu	DPT01
2	S0011	Saranaga	DPT07
3	S0012	Rasil	NULL
4	S0013	Dakshina	DPT02
5	S0014	NULL	NULL
6	S0015	Sanduni	DPT04
7	S0016	NULL	NULL
8	S0017	Praveen	DPT10
9	S0018	Samalka	DPT14
10	S0019	NULL	NULL
11	S0020	Vidun	DPT09
12	S0024	Malith	DPT05
13	S0025	NULL	NULL
14	S0026	Saranaga	DPT07
15	S0027	NULL	NULL
16	S0028	Awantha	DPT16

After that one , SAVEPOINT after_dep_update has met. This statement creates a savepoint "called after dep update" which allows changes to be rolled back to this point in this transaction. It's all about the end of the segment of code.

- Updating Staff Name

```
SET TRANSACTION NAME 'Staff_update';

update staff
SET staff_fname = 'Jagath'
WHERE staff_id = 'S0024';

ROLLBACK;

ROLLBACK TO SAVEPOINT after_the_staff_update;
```

Same as the first example this statement begins a transaction and names it as ‘Dependent_Update’. And After that it retrieve the all data for before the changes to do.

Then it update the Staff_ID of ‘S0024’, the first name as “Jagath”.

	LOGIN_ID	STAFF_ID	STAFF_FNAME	STAFF_LNAME	DATE_OF_BIRTH
1	L0124	S0010	Lahiru	Munasinghe	23-FEB-96
2	L0125	S0011	Eranga	Gimsara	20-DEC-97
3	L0134	S0012	Kavindu	Jayasinghe	12-SEP-98
4	L0135	S0013	Nuwanthi	Madushika	23-OCT-94
5	L0140	S0014	Nimesh	Maduwantha	10-MAR-95
6	L0143	S0015	Jayani	Dilmini	19-JUN-99
7	L0144	S0016	Dinithi	Mohotty	01-JAN-98
8	L0145	S0017	Gethmi	Kaveesha	09-FEB-95
9	L0150	S0018	Chamani	Jayanka	31-OCT-99
10	L0152	S0019	Amila	Rathnayaka	12-FEB-98
11	L0155	S0020	Vinsadi	Sinethma	26-JUL-94
12	L0156	S0024	Jagath	Randima	25-NOV-95
13	L0123	S0025	Jetumini	Imanahari	15-DEC-95
14	L0157	S0026	Yasuri	Bandaara	25-DEC-95
15	L0160	S0027	Apeksha	Warnakulasuriya	15-APR-99
16	L0161	S0028	Hasitha	Suraweera	02-FEB-95

This is the after of the update statement.

But if we just want to redo that statement it has the code statement and called it as **"ROLLEBACK".** Rolling back means undoing any modifications to the data that were made

During an uncommitted transaction through SQL statements. To store old values Oracle uses tablespaces (or rollback segments) to undo. The log redo includes a record of modifications.

Oracle lets you roll back a whole transaction that is uncommitted. Additionally, the trailing portion of an uncommitted transaction can be rolled back to a marker called a savepoint. In that name ROLLBACK TO SAVEPOINT is named "after_the_staff_update."

- Retrieving Data.

The following statements will be executed on a single day-to-day basis to collect the order number to Supermarkets' managers and that report would not be influenced by any other consumer who will add or delete code statements.

```
COMMIT;

SET TRANSACTION READ ONLY NAME 'Review';

SELECT staff_id,number_of_delivery
FROM delivery_person;

COMMIT;
SAVEPOINT after_retrive_DPerson;
```

The first COMMIT declaration guarantees the SET TRANSACTION is the first declaration in the transaction. In addition, the last COMMIT statement does not make any changes to the database permanent. The read-only transaction literally ends.

07. Access control privileges

```
---Access control privileges
---There are four implements----

--Create Roles-
--Grant privileges-
--Create Users-
--Assign User to relevent Role-

-- There are three employees in this DB : Manager, Cashier, Delivery_Person --


--1.Create roles

ALTER SESSION SET "_ORACLE_SCRIPT"=TRUE;

CREATE ROLE Manager;
CREATE ROLE Cashier;
CREATE ROLE Delivery_Person;

--2.Creating Users--

--for manager--
CREATE USER Lahiru IDENTIFIED BY "ytLm123#w1";
CREATE USER Eranga IDENTIFIED BY "k12#aqeftK";
CREATE USER Kavindu IDENTIFIED BY "A424qef#el";

--for cashier--

CREATE USER Nuwanthi IDENTIFIED BY "Lq12dr5&fe";
CREATE USER Nimesh IDENTIFIED BY "KL12w$qmn9";
CREATE USER Jayani IDENTIFIED BY "ewdee#6gt";
CREATE USER Dinithi IDENTIFIED BY "zw$def#6ht";
CREATE USER Gethmi IDENTIFIED BY "awdef#6ht&";
```

```
--for Delivery_Person--

CREATE USER Chamani  IDENTIFIED BY "123ef#ghg&";
CREATE USER Amila IDENTIFIED BY "564$kmql&8";
CREATE USER Vinsadi IDENTIFIED BY "qswemhfl&1";
CREATE USER Sahan IDENTIFIED BY "1234qwvgr&";
CREATE USER Jetumini IDENTIFIED BY "LwseM1264";
CREATE USER Yasuri IDENTIFIED BY "0973jhebrnm";
CREATE USER Apeksha IDENTIFIED BY "degtL12%km";
CREATE USER Hasitha  IDENTIFIED BY "1kTa123J&m";


--To see the name of the databese--

select*from global_name;


--Granting all access to the Manager--
GRANT ALL Privileges TO Manager;

--Granting Select, Insert, Update and Delete privilage--

GRANT CONNECT TO Cashier;
GRANT Select,insert ON Product TO Cashier;
GRANT Select,insert,update,delete ON Payment TO Cashier;
GRANT Select,update ON Login TO Cashier;

GRANT CONNECT TO Delivery_Person;
GRANT Select,insert,update,delete ON Orders TO Delivery_Person;
GRANT Select,update ON Login TO Delivery_Person;
```

```
--Grant Users For Roles--

Grant Manager TO Lahiru;
Grant Manager TO Eranga;
Grant Manager TO Kavindu;

Grant Cashier TO Nuwanthi;
Grant Cashier TO Nimesh;
Grant Cashier TO Jayani;
Grant Cashier TO Dinithi;
Grant Cashier TO Gethmi;

Grant Delivery_Person TO Chamani;
Grant Delivery_Person TO Amila;
Grant Delivery_Person TO Vinsadi;
Grant Delivery_Person TO Sahan;
Grant Delivery_Person TO Jetumini;
Grant Delivery_Person TO Yasuri;
Grant Delivery_Person TO Apeksha;
Grant Delivery_Person TO Hasitha;
```

08. Web application attacks and the countermeasures

- Data security risks - The integrity and privacy of data are at risk from unauthorized users, external sources listening in on the network, and internal users giving away the store.

- Data tampering - Data can be modified or viewed in transit

- Data theft - Highly sensitive data can be stolen, such as credit card numbers of patients, etc.

- Authentication Failure - Web server flaws related to security exist when appropriate user security mechanisms are applied inappropriately. Its danger breaching user accounts. These network protection flaws can be abused by attackers to obtain leverage over every user account or even the system.
    - The Credential Stuffing flaw involves an intruder checking a set of appropriate passwords and usernames gleaned out of a separate attack, before the intruder can recognize a suitable combination and obtain entry.
    - The Brute Force attack, in which the intruder tries every conceivable combination of characters before they reach a suitable one, is another the weakness.
    - Another typical attack due to authentication failure is the hijacking of a session. In this scenario, client IDs cannot be correctly

disallowed, and attackers are able to manipulate a legible user's authenticated client.

- Falsifying user identities - Without a proper authentication for the user base, masquerading and falsification attack can be performed on the distributed system.

- Security Misconfiguration - Security misconfigurations enable attackers to access your website easily and thus render it one of the most important web application vulnerabilities you need to prevent.
    - Many protection vulnerabilities that attackers may manipulate to obtain unauthorized access include empty websites, unpatched data, vulnerable software, and default settings.
    - Each stage will be exposed to protection modifications on the application stack. That covers your site server, your servers, your file server, your computing facilities, your system, your application server, and more.
    - Through leveraging these web server bugs, attackers will obtain private details and monitor the user and administrator accounts.

- Excessive privileges

- Legitimate privilege abuse - Authorized users may abuse the privileges granted to them

- Database injection attack - SQL injection attacks, a very common attack type

- SQL injection
  - A ton of hackers are attempting with SQL injection attacks to obtain entry to the servers. In order to gather data and administer the database, the intruder introduces false SQL statements into the template fields and other entry points. They will modify and change or even destroy the information and assault the fundamental structure with this information.
  - Attackers usually capture important consumer information including their contact numbers, passwords or even credit card information through these assaults. These vulnerability flaws can often be taken advantage of, for example, to adjust the price of a drug. Advanced attacks can also enable the server and the operating system to be managed.
    - ✓ Example
      - In a username or a password area, an intruder may position the malicious SQL query. On the server side the application will be conducted in an inappropriate database operation if validations for this information are not done. This attack can contribute to abuse of the database, the extraction of confidential company information or the privacy of customers.

- Malware attack

- XML External Entities - The unintended XML intrusion or an XML manipulation assault is some type of weakness you will be aiming into. Every XXE bug is a weakness. Such kinds of assaults arise when attackers utilize an XML parser that

is weakly installed. By means of these attacks, attackers may insert additional details, access sensitive data, run programs, and build shells.

- The remote execution of code, the Side Request Forging Service, and more can result in XML attacks by external entity. Most XML parsers are susceptible to these attacks by nature. This helps developers to guarantee that their web application is clear of such vulnerabilities.

- Storage media exposure

- Broken Access Control (Authorization Failure) - Access management lets you manage which page pages and which apps may be viewed by specific tourists.
    - When, for example, the website is a multi-seller portal to display their goods, they need some connections to introduce new items to handle their selling. But not all visitors are expected to have such access as most visitors purchase items on your website.
    - Therefore, a compromised access control opens up loopholes in the web that can be abused by criminals to access private details or unauthorized features. You can also allow improvements to access privileges and user details utilizing such assaults.

- The exploitation of vulnerable misconfigured database.

- DoS and DDoS attacks - short for Distributed Denial of Service.DDoS is a type of DOS attack where multiple compromised systems, which are often infected with a Trojan, are used to target a single system causing a denial of Service (DoS) attack.

- Cross site request forgery (CSRF) -This is a form of attack that triggers an unintended behavior by the Web client in the program to which the user connects.

    - ✓ Example
        - A suspect may write a deceptive script that leads to some money being transferred to his account and wraps it in an harmless, look hyperlink. This connection is then spread to other bank clients. If you click on this link when you sign into your bank account, the transfer should start accidentally.

- Cross site scripting (XXS) - As already stated, cross-site scripting or XSS is a web application weakness that could threaten the protection of your users. Such assaults insert and execute malicious code on the client-side into the operating program.
    - The purpose of XSS attacks is to transmit this malicious code to other users, often to the malware or to capture confidential information from their computers. This website vulnerabilities will cause the user's web browser to be fully managed and incredibly dangerous for any website.
        - ✓ Example
            - The injected malicious code is run as the corresponding page loads it. Attackers may take disruptive steps such as covering a password, user history access and so on.

Preventive Measures to Mitigate Attacks

- To mitigate these attacks there are some preventive measures that can be incorporated with applications, like:

    - Web application firewall (WAF) -WAFs provide a firewall between the browser and the web client. It filters the site server http requests based on those predefined policies. The code usually defends against such threats as SQL Injection, cross-site scripting, etc.

    - Cryptography - Convert data into unreadable format to unwanted object using encryption and decoding methods. Information cannot be read without a key to decode it.

- SQL injection-

    - Prepared database statements will reduce web server weaknesses relevant to SQL. A prepared statement helps to sanitize the input and makes sure it is not part of the SQL query but a string literally in SQL. In other terms, the database can differentiate between SQL and SQL code. The application is no longer prone to attacks by SQL injection because it is less vulnerable to falsification.

    - Another excellent alternative is moving to entity relativity mapping (ORMs). In addition to parametrized queries, most ORMs support non-parametric queries. It is therefore necessary to take proper note of frameworks.

- Make the most of LIMIT and other SQL controls inside your queries to avoid the mass disclosure of records even if a SQL injection attack does occur.

- Cross site scripting (XXS)-
  - Modern frameworks have made the escape of untrusted user feedback and prevention of XSS attacks much easier. AngularJS, React JS and Ruby on Rails are some of the new, most powerful frameworks built to avoid such vulnerabilities in web applications. While they have drawbacks, these frameworks can automatically avoid user feedback and help mitigate XSS attacks by design.

  - Avoid enforcing a blacklist, rather than a whitelist, since blacklists are less efficient in avoiding vulnerabilities in Web security. An attacker who knows what they are doing will easily bypass a filter in the blacklist.

  - The ultimate solution to avoid such vulnerabilities in web applications is encoding the data. It includes translating untrusted user input into a secure form such that the information is presented as data to the user without executing it as code in the browser. It means that unique characters are converted into an identical form that the user won't consider important any longer.

  - It is also important to understand that the encoding of output depends on the context of where data is being generated. For

example, you might have HTML contexts, HTML entity contexts, HTML context attributes, CSS contexts, JavaScript contexts, and more. As such, when making the browser page you will need to apply context sensitive encoding.

- Allow a Content Protection Policy (CSP), which can be very effective in mitigating vulnerabilities in Cross-Site Scripting.

- Authentication Failure –
  - One of the important steps to prevent such bugs in the web application is to give developers ample time to check the code before it is deployed for production. External compliance assessments will also help ensure you are implementing website compliance best practices.
  - Avoid deploying credentials by default, particularly for admins.
  - Make sure you enforce multi-factor authentication whenever possible to make your device less vulnerable to the attacks listed above.
  - Put a cap or delay on failed attempts to log in. Make sure you report all of the faults and alert administrators when an attempted attack happens.

- Evite restricting input size unnecessarily. If you require more characters, attackers would have less chances of guessing the correct password.

- Have some form of lockout set up to prevent brute force attacks and mitigate these vulnerabilities in web applications.

- Using adaptive hashing algorithms such as bcrypt, pbkdf2, argon2, etc. to salt and hash passwords before saving them to the database.

- Implement poor password checks to boost the protection of the password. This will include checking and comparing new or modified passwords against a list of compromised or poor passwords. Using a tool to search for compromised passwords (such as Have I Been Pwned) helps simplify this feature and keeps up to date the list of compromised passwords as new attacks take place.

- Security Misconfiguration –
    - Make sure you use authenticated links (HTTPS) to transfer information and data between users and the device.

    - Have a repeatable hardening procedure that can be implemented in another environment quickly and easily. This saves time in creating a new and healthy environment, as you can automate the process.

- Remote administrative tasks can be conducted through secure channels such that vulnerabilities are minimized. Even if you are using non-strong encryption protocols, ensure that they are disabled over a secondary encryption channel such as IPSEC or TLS.

- Perform routine file completeness checks to ensure that no unauthorized changes have been made to sensitive files. Using file integrity testing software to accept normal and expected improvements while alerting you of unintended or irregular modifications.

- Develop an automatic method to test the settings and parameters in all applications daily. You can use an automated configuration monitoring tool to alert you to unauthorized modifications. This will help you recognize the flaws in these online applications before harm is done.

- Hold the infrastructure small to discourage unwanted apps, tests, documentation to components from being added. When you have unspent platform components and objects, the easiest way to disable them is to avoid bugs in software applications.

- XML External Entities –
    - The most reliable means of preventing XXE attacks is to completely deactivate Document Form Definitions (DTDs), also

known as External Entities. This means that the analyzer does not attack. DTDs cannot all be activated entirely, though. In this case, for each parser, you need to delete statements of the form of external text and external entities.

- You will try to use less complicated data formats such as JSON as far as possible. It is also important that confidential data will not be serialized to prevent bugs in such database applications.

- To verify, optimize, and filter data, apply a constructive server-side process. This helps deter aggressive data from happening within the XML records, nodes and/or headers to avoid web server bugs associated with XXE.

- Update or redesign all XML processors and libraries used by the program or OS.

- While manual code analysis in broad and complicated applications is a great way to use crucial features, you can also use SAST tools to identify XXE in source code.

- To check an incoming XML form, using XSD validation or an analogous substitute.

- Broken Access Control (Authorization Failure) –
    - In designing and configuring applications, it is important to establish a first security strategy.

- Refuse entry by default except for public services.

- Ensure the security is extended both horizontally (all data) and vertically (all access privileges). Vertical security requires the use of the least privileged definition where access is only given according to the present functions and obligations and no more.

- Centralize all judgment authorization in order to eliminate web application risks relevant to access.

- Using Template Access Controls to implement database control rather than allowing users to build, read, edit or erase any data. Disable the right of other users to read or change data.

- Implement a system of one-time access control that the program will reuse.

- Set API and access constraints to reduce the chance of automated device attacks.

- Invalidate JWT tokens and user sessions on the server after logout.

- Allow the display of web server folders and prohibit web root storage of backup and metadata files too.

09. Countermeasures implementation method and security mechanisms.

- Retrieve unauthorized statements from other users.

All database data should be accessed by the approved person of the database, Ex: Employee can not access login credentials from the login table. He will only be able to view login IDs and names, not credentials.

Here is a code statement to that effect,

```
-- Set up Fred's account - you need CREATEROLE privilege for this!
CREATE USER Gunadasa IDENTIFIED BY "qemhf171";
 -- Prevent Fred from reading the base table
REVOKE ALL PRIVILEGES ON login FROM Gunadasa;
 -- Create the view that contains what Fred can access
CREATE VIEW PublicLoginViews AS SELECT  login_id ,login_name FROM login;
 -- Allow Fred to read the view
GRANT SELECT ON PublicLoginViews TO Gunadasa;
```

Script Output

Task completed in 0.197 seconds

```
Revoke succeeded.

View PUBLICLOGINVIEWS created.

Grant succeeded.
```

After this implementation, the user of "Gunadasa" will be allowed to see only the login name as well as login ID columns of the login table.

- Creating an Oracle trigger to back-up the data.

Suppose that client needs to report conduct towards the staff table when modifying or deleting an employee. To do this: Then, create a new table to record events UPDATE and DELETE for backup.

Worksheet Query Builder

```
CREATE TABLE audits (
        audit_id         NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
        table_name       VARCHAR2(255),
        transaction_name VARCHAR2(10),
        by_user          VARCHAR2(30),
        transaction_date DATE
);
```

Script Output Query Result Query Result 1

```
Table AUDITS created.
```

Then create a new trigger associated with the customers table after that table has been created:

```
CREATE OR REPLACE TRIGGER employee_audit_trg
    AFTER
    UPDATE OR DELETE
    ON login
    FOR EACH ROW
DECLARE
   l_transaction VARCHAR2(10);
BEGIN
   -- determine the transaction type
   l_transaction := CASE
         WHEN UPDATING THEN 'UPDATE'
         WHEN DELETING THEN 'DELETE'
   END;

   -- insert a row into the audit table
   INSERT INTO audits (table_name, transaction_name, by_user, transaction_date)
   VALUES('Login', l_transaction, USER, SYSDATE);
END;
/
```

Script Output | Query Result | Query Result 1

Task completed in 0.09 seconds

```
Table AUDITS created.

Trigger EMPLOYEE_AUDIT_TRG compiled
```

The following clause:

```
ALTER TRIGGER employee_audit_trg ENABLE;
```

Trigger will allow database after the following code.

And the trigger will be fired after a row in the client table has been updated or deleted.

Anyone will decide the actual behavior within the trigger whether it's UPDATE or DELETE and insert a row into the audit table.

- When creating more sensitive data table, It should has more security than other tables.

Like the login table, all users' passwords should always be encrypted because of their sensitive data.

Before that, it is necessary to create more security tables to insert sensitive data.

Worksheet | Query Builder

```
Create Table Login(
    login_ID char(10) Not Null,
    login_Name varchar(25) Not Null,
    login_password char(10) Not Null
);

ALTER TABLE Login ADD(CONSTRAINT login_users_pk PRIMARY KEY (login_ID));
ALTER TABLE Login ADD(CONSTRAINT login_users_uk UNIQUE (login_Name));
CREATE SEQUENCE login_users_seq;
```

Script Output | Query Result | Query Result 1 | Query Result 2

Task completed in 0.106 seconds

```
Table LOGIN altered.


Table LOGIN altered.


Sequence LOGIN_USERS_SEQ created.
```

Next, create a package that contains the security code specification.

Worksheet | Query Builder

```
CREATE OR REPLACE PACKAGE login_user_security AS

  FUNCTION get_hash (p_uname  IN  VARCHAR2,
                     p_pword  IN  VARCHAR2)
    RETURN VARCHAR2;

  PROCEDURE add_user (p_uname  IN  VARCHAR2,
                      p_pword  IN  VARCHAR2);

  PROCEDURE change_pword   (p_uname      IN  VARCHAR2,
                            p_old_pword  IN  VARCHAR2,
                            p_new_pword  IN  VARCHAR2);

  PROCEDURE valid_user (p_uname  IN  VARCHAR2,
                        p_pword  IN  VARCHAR2);

  FUNCTION valid_user (p_uname  IN  VARCHAR2,
                       p_pword  IN  VARCHAR2)
    RETURN BOOLEAN;

END;
/
```

Script Output | Query Result | Query Result 1 | Query Result 2

Task completed in 0.122 seconds

```
Package LOGIN_USER_SECURITY compiled
```

Create a package body to define the actual operations after that.

```
CREATE OR REPLACE PACKAGE BODY login_user_security AS

  FUNCTION get_hash (p_uname  IN  VARCHAR2,
                     p_pword  IN  VARCHAR2)
    RETURN VARCHAR2 AS
    l_salt VARCHAR2(30) := 'PutYourSaltHere';
  BEGIN
    RETURN DBMS_OBFUSCATION_TOOLKIT.MD5(
    input_string => UPPER(p_uname) || l_salt || UPPER(p_pword));
  END;

  PROCEDURE add_user (p_uname  IN  VARCHAR2,
                      p_pword  IN  VARCHAR2) AS
  BEGIN
    INSERT INTO Login (
      login_ID,
      login_Name,
      login_password
    )
    VALUES (
      login_users_seq.NEXTVAL,
      UPPER(p_uname),
      get_hash(p_uname, p_pword)
```

Script Output

Task completed in 0.139 seconds

```
Package Body LOGIN_USER_SECURITY compiled
```

Worksheet Query Builder

```
    VALUES (
      login_users_seq.NEXTVAL,
      UPPER(p_uname),
      get_hash(p_uname, p_pword)
    );

    COMMIT;
  END;

  PROCEDURE change_pword (p_uname      IN  VARCHAR2,
                          p_old_pword  IN  VARCHAR2,
                          p_new_pword  IN  VARCHAR2) AS
    v_rowid  ROWID;
  BEGIN
    SELECT rowid
    INTO   v_rowid
    FROM   Login
    WHERE  login_Name = UPPER(p_uname)
    AND    login_password = get_hash(p_uname, p_old_pword)
    FOR UPDATE;

    UPDATE Login
    SET    login_password = get_hash(p_uname, p_new_pword)
    WHERE  rowid    = v_rowid;
```

Script Output

Task completed in 0.139 seconds

```
Package Body LOGIN_USER_SECURITY compiled
```

```
  COMMIT;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20000, 'User name or Password Wrong');
END;

PROCEDURE valid_user (p_uname  IN  VARCHAR2,
                      p_pword  IN  VARCHAR2) AS
  v_dummy  VARCHAR2(1);
BEGIN
  SELECT '1'
  INTO   v_dummy
  FROM   Login
  WHERE  login_Name = UPPER(p_uname)
  AND    login_password = get_hash(p_uname, p_pword);
EXCEPTION
  WHERE NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20000, 'User name or Password Wrong');
END;

FUNCTION valid_user (p_uname  IN  VARCHAR2,
                     p_pword  IN  VARCHAR2)
  RETURN BOOLEAN AS
BEGIN
  valid_user(p_uname, p_pword);
```

Script Output

Task completed in 0.139 seconds

```
Package Body LOGIN_USER_SECURITY compiled
```

```
  BEGIN
    SELECT '1'
    INTO   v_dummy
    FROM   Login
    WHERE  login_Name = UPPER(p_uname)
    AND    login_password = get_hash(p_uname, p_pword);
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR(-20000, 'User name or Password Wrong');
  END;

  FUNCTION valid_user (p_uname  IN  VARCHAR2,
                       p_pword  IN  VARCHAR2)
    RETURN BOOLEAN AS
  BEGIN
    valid_user(p_uname, p_pword);
    RETURN TRUE;
  EXCEPTION
    WHEN OTHERS THEN
      RETURN FALSE;
  END;

END;
/
```

Script Output

Task completed in 0.139 seconds

```
Package Body LOGIN_USER_SECURITY compiled
```

The overloads of VALID USER allow the security check to be performed in a different way.

The function GET HASH is used to hash user name and password combinations. It always returns a VARCHAR2(16) irrespective of the length of the input parameters. This compression level means that the hash value may not be unique, hence the unique constraint on the USERNAME column.

The DBMS UTILITY.GET HASH VALUE function could be used to replace the DBMS OBFUSCATION TOOLKIT.MD5 function, but the former hash algorithm is not guaranteed to remain constant between database versions.

Testing

First create a new user. And when it implement, It has like below output,

```
SQL> exec app_user_security.add_user('tim','MyPassword');

PL/SQL procedure successfully completed.

SQL> select * from app_users;

        ID USERNAME                       PASSWORD
---------- ------------------------------ ----------------
         1 TIM                            [w◆44Z◆◆8fE◆◆
```

10. The ways for apply the recovery mechanisms to database.

Recovery of the data base is the method of restoring the database to the most stable state of justification that existed just before the failure. Database recovery is achieved in three phases.

1. Pre-condition: The database is in a coherent state at any given time.
2. Condition: Some kind of system failure occurs.
3. Post-condition: Restore the database to consistent pre-failure level.

There are both automatic and non-automatic ways to recover from a circumstance of failure. Among certain purposes (system crash, transaction errors, malware, catastrophic failure, incorrect execution of commands) the techniques used to retrieve the missing data are called as techniques among data base recovery.

Recovery strategies rely on providing a special file known as a device log. It includes information about each transaction's start and end, and any changes that occur in the transaction. The log keeps track of all transaction operations which affect data base item values. To recover from transaction failure this information is needed.

Undoing – If a transaction fails, then the recovery manager will undo transactions (reverse transaction operations). This includes inspecting a transaction for the write item(T, x, old value, new value) log entry and setting to old-value the value of item x in the database. There are two main strategies for recovering from non-catastrophic transaction failures. Those are pending changes and upcoming changes.

Deferred update – This method does not update the database physically on the disk until a transaction has reached its point of commit. All transaction updates are registered in the workspace of the local transaction before the commit is reached. If a transaction fails before it reaches its commit stage, the database would not have been updated in any way so that UNDO is not required. The effect of the operations reported in the local transaction workspace will need to be REDO, as their effect may not have been written in the database yet. Hence a delayed update is also known as the algorithm No-undo / redo.

Immediate update-In the immediate update, some transaction operations that update the database before the transaction reaches its commit point. However, these operations are documented in a disk log before being added to the database, which also allows recovery possible. When a transaction fails to hit its commit stage, it must reverse the effect of its operation. This is known as the undo / redo algorithm.

Caching / Buffering – One or more disk pages containing modified data objects are cached into main memory buffers and then restored to memory until they are written back to disk. For keeping those buffers, a list of in-memory buffers called the DBMS cache is held under DBMS power. A directory is used to keep track of which objects in the buffer

are in the database. Through buffer is associated with a dirty bit, which is 0 if the buffer is not updated rather than 1 if changed.

Shadow paging-Atomicity and durability are given. A directory is constructed with n entries, where the ith entry points to the connection page of the ith database. When a transaction starts to execute the current directory will be copied to a shadow directory. If a page is to be changed, a shadow page is allocated where changes are made and all pages that refer to the original are revised to refer to the replacement page when it's ready to become permanent.

Each such entry contains a pointer to a disk page. Suppose a transaction exists to perform a writing operation that resides on the selected page. If the page is not in the main memory already then the program must issue an request. And, if the write first performed on the page whose transaction is to be performed, the program will change the current page to locate the unused page on the disk, then it will remove the (unused) page previously found from the free page frames list, and copy the content to the unused page in question. Then the current page table is updated and values are allocated in the buffer.

Using this form of technique for data base recovery can easily recover the databases from issues.