SQL QUERIES

QUERY 1:

1. **User**(s): Owner

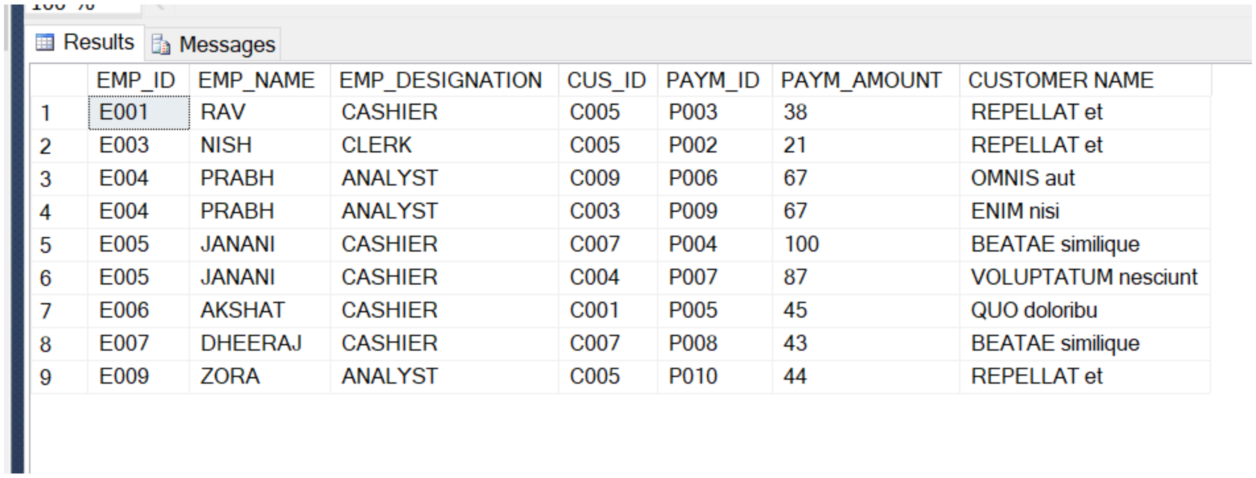
2. **Purpose**: Owner can check what all payments have been processed by any employee for any customer. It can be customised to put a restriction on where clause. For e.g: restrict employee id = E001 to check all payments processed by one employee or it can use where payment BEWTEEN 1 AND 100 to check all payment details done by all employees with values in range 1 to 100.

3. **Required layout for the result set:** Attributes displayed include attributes employee id, employee name, employee designation, customer id, payment id, payment amount, customer name.

4. **SQL statement**

```
SELECT     E.EMP_ID,
          E.EMP_NAME,
          E.EMP_DESIGNATION,
           C.CUS_ID,
          P.PAYM_ID,
          P.PAYM_AMOUNT,
         (UPPER(CUS_FNAME)+' '+CUS_LNAME) AS 'CUSTOMER NAME'

FROM
              EMP AS E JOIN PAYMENT AS P
ON            E.EMP_ID = P.EMP_ID
              JOIN CUST AS C
ON            P.CUS_ID = C.CUS_ID

WHERE         PAYM_AMOUNT BETWEEN 1 AND 100
```

5. **Result set generated by query in database**



	EMP_ID	EMP_NAME	EMP_DESIGNATION	CUS_ID	PAYM_ID	PAYM_AMOUNT	CUSTOMER NAME
1	E001	RAV	CASHIER	C005	P003	38	REPELLAT et
2	E003	NISH	CLERK	C005	P002	21	REPELLAT et
3	E004	PRABH	ANALYST	C009	P006	67	OMNIS aut
4	E004	PRABH	ANALYST	C003	P009	67	ENIM nisi
5	E005	JANANI	CASHIER	C007	P004	100	BEATAE similique
6	E005	JANANI	CASHIER	C004	P007	87	VOLUPTATUM nesciunt
7	E006	AKSHAT	CASHIER	C001	P005	45	QUO doloribu
8	E007	DHEERAJ	CASHIER	C007	P008	43	BEATAE similique
9	E009	ZORA	ANALYST	C005	P010	44	REPELLAT et

QUERY 2:

1. **User**(s): Employee (cashier, analyst, clerk etc)

2. **Purpose**: Employee can check which all reservation has been done successfully for any corresponding payment done by a specific customer. It can be customised to put a restriction on where clause.

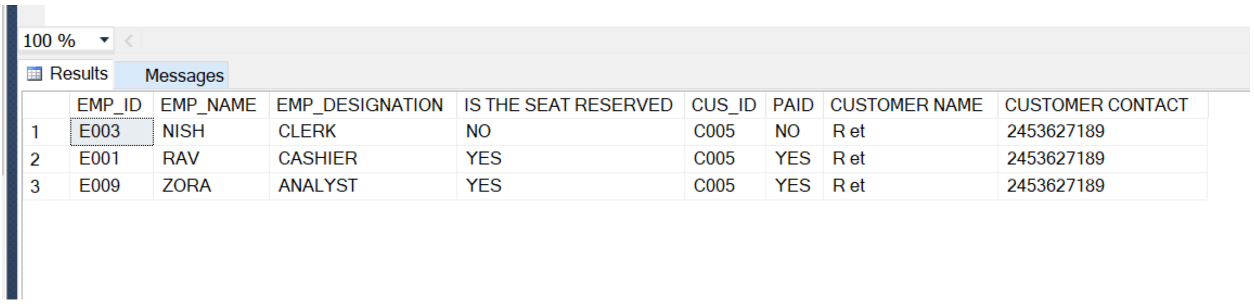
For e.g: restrict employee id = E001 to check all payments processed by one employee to see whether seat has been reserved or not. In our query we have used customer last name = 'et' as an example. It can use wildcards to extend usage to find for any customer last name and see his multiple reservations and payment confirmation.

3. **Required layout for the result set**: Attributes displayed include attributes employee id, employee name, employee designation, IS the seat reserved , customer id, paid confirmation column, customer name and customer contact.

4. **SQL statement**

```
SELECT    E.EMP_ID,
          E.EMP_NAME,
          E.EMP_DESIGNATION,
          R.RESERVED AS 'IS THE SEAT RESERVED',
          C.CUS_ID,R.PAID,

          (UPPER(SUBSTRING(C.CUS_FNAME,1,1))+ ' ' +C.CUS_LNAME) AS 'CUSTOMER NAME',
          C.CUS_PHNO AS 'CUSTOMER CONTACT'
FROM      EMP AS E    JOIN RESERVATION AS R ON E.EMP_ID = R.EMP_ID
                     JOIN CUST AS C ON C.CUS_ID = R.CUS_ID

WHERE    C.CUS_LNAME LIKE 'et'

ORDER BY E.EMP_NAME
```

5. **Result set generated by query in database**

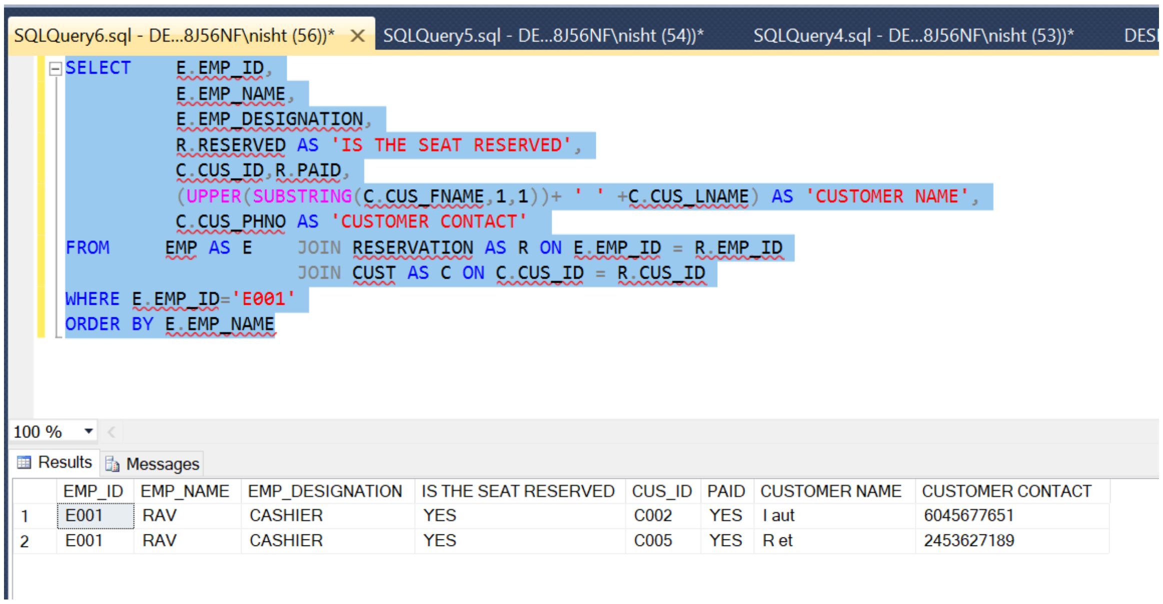


100 %

Results | Messages

	EMP_ID	EMP_NAME	EMP_DESIGNATION	IS THE SEAT RESERVED	CUS_ID	PAID	CUSTOMER NAME	CUSTOMER CONTACT
1	E003	NISH	CLERK	NO	C005	NO	R et	2453627189
2	E001	RAV	CASHIER	YES	C005	YES	R et	2453627189
3	E009	ZORA	ANALYST	YES	C005	YES	R et	2453627189

Option 2:


SQLQuery6.sql - DE...8J56NF\nisht (56))*
SQLQuery5.sql - DE...8J56NF\nisht (54))*
SQLQuery4.sql - DE...8J56NF\nisht (53))*
100 %
Results
Messages

```
SELECT      E.EMP_ID,
            E.EMP_NAME,
            E.EMP_DESIGNATION,
            R.RESERVED AS 'IS THE SEAT RESERVED',
            C.CUS_ID,R.PAID,
            (UPPER(SUBSTRING(C.CUS_FNAME,1,1))+ ' ' +C.CUS_LNAME) AS 'CUSTOMER NAME',
            C.CUS_PHNO AS 'CUSTOMER CONTACT'
FROM       EMP AS E      JOIN RESERVATION AS R ON E.EMP_ID = R.EMP_ID
                         JOIN CUST AS C ON C.CUS_ID = R.CUS_ID
WHERE E.EMP_ID='E001'
ORDER BY E.EMP_NAME
```

	EMP_ID	EMP_NAME	EMP_DESIGNATION	IS THE SEAT RESERVED	CUS_ID	PAID	CUSTOMER NAME	CUSTOMER CONTACT
1	E001	RAV	CASHIER	YES	C002	YES	I aut	6045677651
2	E001	RAV	CASHIER	YES	C005	YES	R et	2453627189

<u>Query 3:</u>

1. **User**(s): Customer

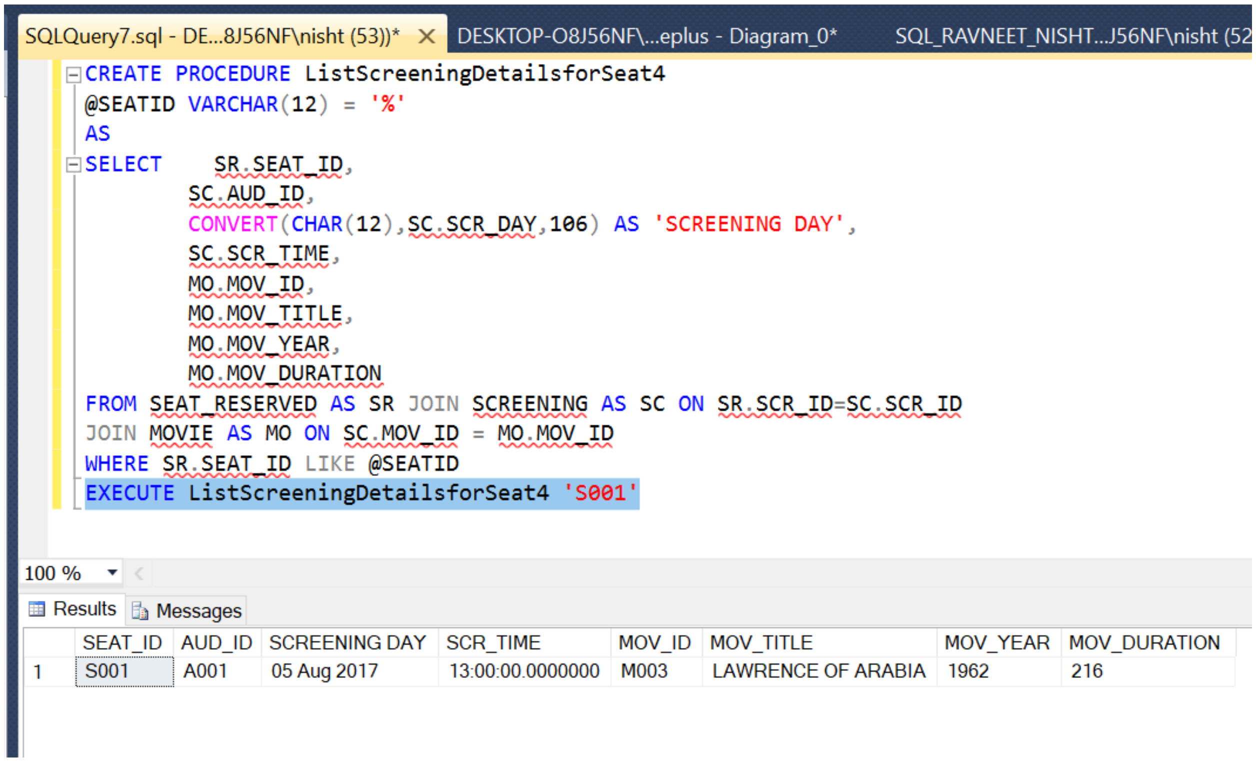
2. **Purpose**: This query creates a procedure which takes one input parameter from customer which is seat id. For any seat provided by customer, entire screening details such as seat id, auditorium id the movie id, title, year, duration and screening day and time can be retrieved by any customer for his particular seat.

3. **Required layout for the result set:** Attributes displayed include SEAT ID, AUDITORIUM ID, SCREENING DAY in reformatted value, screening time, movie id, movie title, movie year and movie duration. We have joined SEAT_RESERVED, SCREENING & MOVIE tables.

4. **SQL statement**

```
CREATE PROCEDURE ListScreeningDetailsforSeat4

@SEATID VARCHAR(12) = '%'
AS

SELECT  SR.SEAT_ID,
        SC.AUD_ID,
        CONVERT(CHAR(12), SC.SCR_DAY,106) AS 'SCREENING DAY',
        SC.SCR_TIME,
        MO.MOV_ID,
        MO.MOV_TITLE,
        MO.MOV_YEAR,
        MO.MOV_DURATION

FROM    SEAT_RESERVED AS SR JOIN SCREENING AS SC ON SR.SCR_ID=SC.SCR_ID
                            JOIN MOVIE AS MO ON SC.MOV_ID = MO.MOV_ID

WHERE    SR.SEAT_ID LIKE @SEATID

EXECUTE    ListScreeningDetailsforSeat4 'S001'
```

5. **Result set generated by query in database**


SQLQuery7.sql - DE...8J56NF\nisht (53))* ✕ DESKTOP-O8J56NF\...eplus - Diagram_0* SQL_RAVNEET_NISHT...J56NF\nisht (52

```
CREATE PROCEDURE ListScreeningDetailsforSeat4
@SEATID VARCHAR(12) = '%'
AS
SELECT     SR.SEAT_ID,
        SC.AUD_ID,
        CONVERT(CHAR(12),SC.SCR_DAY,106) AS 'SCREENING DAY',
        SC.SCR_TIME,
        MO.MOV_ID,
        MO.MOV_TITLE,
        MO.MOV_YEAR,
        MO.MOV_DURATION
FROM SEAT_RESERVED AS SR JOIN SCREENING AS SC ON SR.SCR_ID=SC.SCR_ID
JOIN MOVIE AS MO ON SC.MOV_ID = MO.MOV_ID
WHERE SR.SEAT_ID LIKE @SEATID
EXECUTE ListScreeningDetailsforSeat4 'S001'
```

100 %

Results | Messages

	SEAT_ID	AUD_ID	SCREENING DAY	SCR_TIME	MOV_ID	MOV_TITLE	MOV_YEAR	MOV_DURATION
1	S001	A001	05 Aug 2017	13:00:00.0000000	M003	LAWRENCE OF ARABIA	1962	216

QUERY 4:

1. **User**(s): Owner /Employee

2. **Purpose**: Owner or employee can see for any particular movie , the customer and reservation id and only the seats which have been reserved successfully. Confirmed reservations can be viewed for all movies at once by removing the where clause.

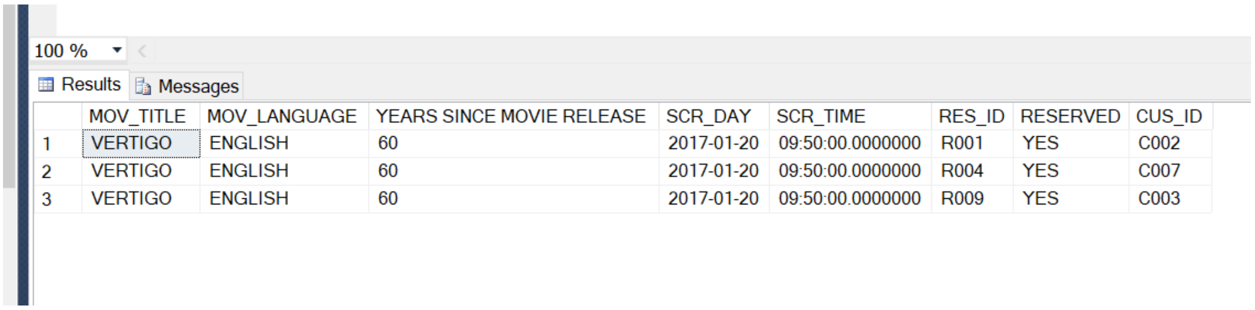
3. **Required layout for the result set:** Attributes displayed include movie title, movie language, years since movie release, screening day, screening time, reservation id, reserved confirmation as yes or no, customer id.

4. **SQL statement**

```
SELECT  M.MOV_TITLE,
        M.MOV_LANGUAGE,
        DATEDIFF(YEAR,M.MOV_DATE_RELEASE,GETDATE()) AS 'YEARS SINCE MOVIE RELEASE',
        SCR.SCR_DAY,
        SCR.SCR_TIME,
        RES.RES_ID,
        RES.RESERVED,
        RES.CUS_ID

FROM RESERVATION AS RES JOIN SEAT_RESERVED AS SR ON RES.RES_ID= SR.RES_ID
```

```
                        JOIN SCREENING AS SCR ON SCR.SCR_ID = SR.SCR_ID
                        JOIN MOVIE AS M ON M.MOV_ID=SCR.MOV_ID

WHERE M.MOV_TITLE LIKE 'VERTIGO' AND RES.RESERVED LIKE 'YES'
```

5. **Result set generated by query in database**



100 %

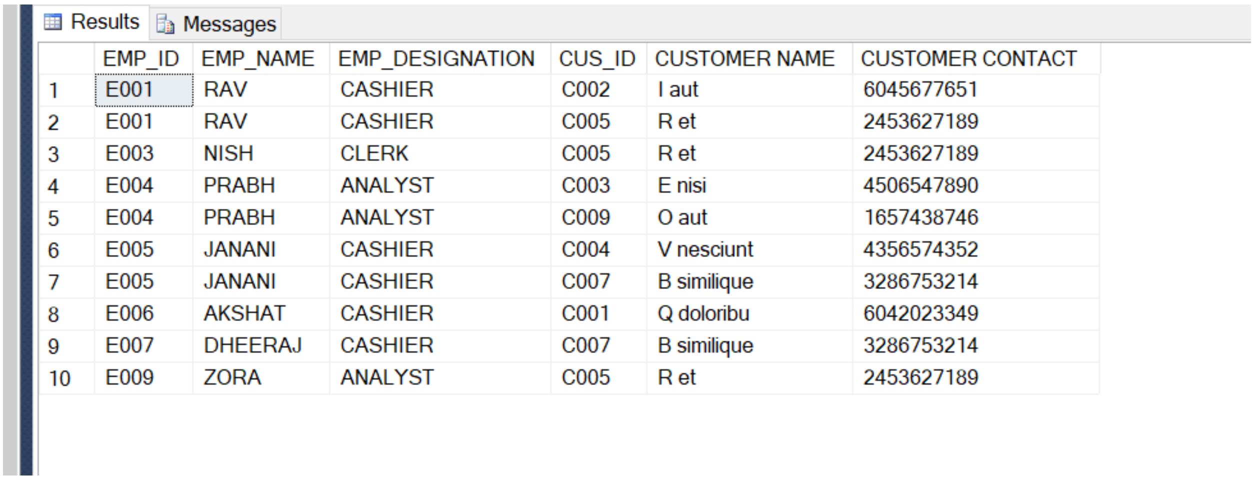
Results | Messages

	MOV_TITLE	MOV_LANGUAGE	YEARS SINCE MOVIE RELEASE	SCR_DAY	SCR_TIME	RES_ID	RESERVED	CUS_ID
1	VERTIGO	ENGLISH	60	2017-01-20	09:50:00.0000000	R001	YES	C002
2	VERTIGO	ENGLISH	60	2017-01-20	09:50:00.0000000	R004	YES	C007
3	VERTIGO	ENGLISH	60	2017-01-20	09:50:00.0000000	R009	YES	C003

<u>Views</u>

<u>**View 1:**</u> The view created provides secured access to the employee data to the customer. The customer can view the employee details without accessing the employee's username and password which are confidential and are known only to the employee.

```
CREATE VIEW EMPLOYEEINFO
AS
SELECT E.EMP_ID,
              E.EMP_NAME,
              E.EMP_DESIGNATION,
              C.CUS_ID,
              (UPPER(SUBSTRING(C.CUS_FNAME,1,1))+ ' ' +C.CUS_LNAME) AS 'CUSTOMER NAME',
              C.CUS_PHNO AS 'CUSTOMER CONTACT'
FROM EMP AS E JOIN RESERVATION AS R ON E.EMP_ID = R.EMP_ID
                          JOIN CUST AS C ON C.CUS_ID = R.CUS_ID

SELECT * FROM EMPLOYEEINFO
```

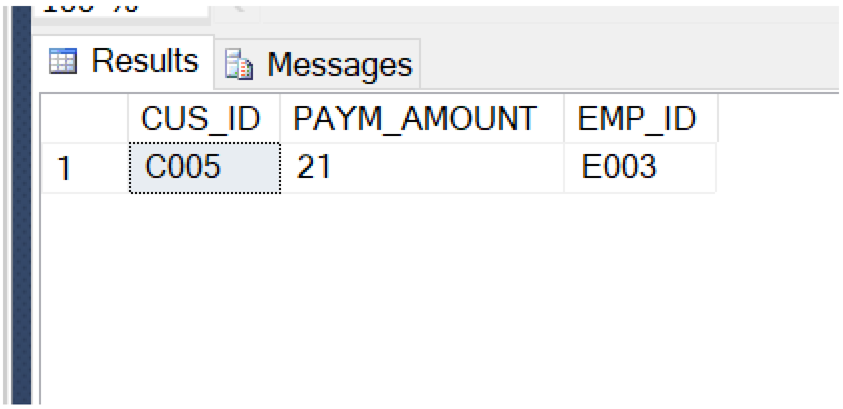


Results | Messages

	EMP_ID	EMP_NAME	EMP_DESIGNATION	CUS_ID	CUSTOMER NAME	CUSTOMER CONTACT
1	E001	RAV	CASHIER	C002	I aut	6045677651
2	E001	RAV	CASHIER	C005	R et	2453627189
3	E003	NISH	CLERK	C005	R et	2453627189
4	E004	PRABH	ANALYST	C003	E nisi	4506547890
5	E004	PRABH	ANALYST	C009	O aut	1657438746
6	E005	JANANI	CASHIER	C004	V nesciunt	4356574352
7	E005	JANANI	CASHIER	C007	B similique	3286753214
8	E006	AKSHAT	CASHIER	C001	Q doloribu	6042023349
9	E007	DHEERAJ	CASHIER	C007	B similique	3286753214
10	E009	ZORA	ANALYST	C005	R et	2453627189

View 2: This view ensures the security of the customer's data from being viewed by any unauthorized employee, like in the given example, cashier is not allowed to access the details of the customers i.e. customer name and phone number. While the analyst is given the authority to view the customers details.

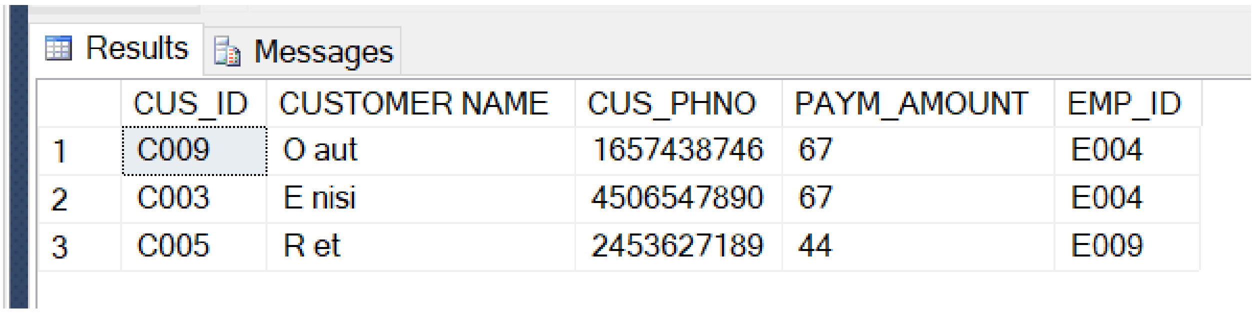
```
    A.  CREATE VIEW CLERKACCESS
AS
SELECT C.CUS_ID,
              P.PAYM_AMOUNT,
              E.EMP_ID
FROM EMP AS E JOIN PAYMENT AS P ON E.EMP_ID = P.EMP_ID
                                                              JOIN CUST AS C ON C.CUS_ID=
P.CUS_ID
WHERE E.EMP_DESIGNATION = 'CLERK'

SELECT * FROM CLERKACCESS
```



Results | Messages

	CUS_ID	PAYM_AMOUNT	EMP_ID
1	C005	21	E003

```
    B.  CREATE VIEW ANALYSTACCESS
AS
SELECT C.CUS_ID,
              (UPPER(SUBSTRING(C.CUS_FNAME,1,1))+ ' ' +C.CUS_LNAME) AS 'CUSTOMER NAME',
              C.CUS_PHNO,
              P.PAYM_AMOUNT,
              E.EMP_ID
FROM EMP AS E JOIN PAYMENT AS P ON E.EMP_ID = P.EMP_ID
                                                              JOIN CUST AS C ON C.CUS_ID=
P.CUS_ID
WHERE E.EMP_DESIGNATION = 'ANALYST'

SELECT * FROM ANALYSTACCESS
```



Results | Messages

	CUS_ID	CUSTOMER NAME	CUS_PHNO	PAYM_AMOUNT	EMP_ID
1	C009	O aut	1657438746	67	E004
2	C003	E nisi	4506547890	67	E004
3	C005	R et	2453627189	44	E009

INDEX

This index is created on reservation table, to check what payment is confirmed and is done by which employee and corresponds to the reservation of which customer.

```
CREATE INDEX RES_INDEX ON RESERVATION (EMP_ID, CUS_ID,PAID);
```