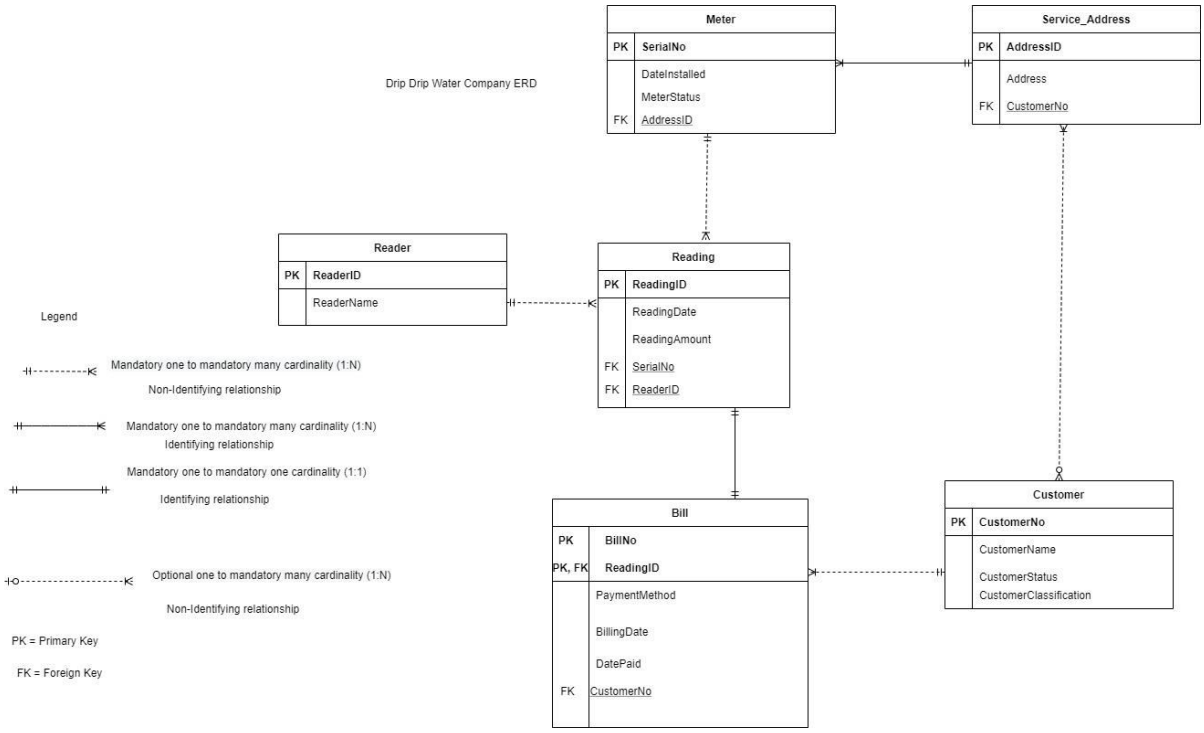
Jin Cherng Chong (33170193) ICT285 Assignment 2

1)



2)

There were several changes I made with my original ERD. Firstly, I added a CustomerStatus attribute in the Customer entity to represent whether the customer has any outstanding bill. This change supports the restriction of not allowing customers with outstanding bills to create new accounts. Secondly, I added a separate Service_Address entity rather than have an address attribute in the meter entity. This makes sense since a single address can have multiple customers over the lifetime of the database. Thirdly, based on feedback I removed the link between Meter and Customer entity as well as renaming some of my current entities for improved readability.

3)

Table: Customer

Attribute name	**Description**	**Any default values?**	**Required?**	**Allowable values**	**Unique?**	**Key?**	**Oracle data type/size**
CustomerNo	Unique Identifier	No	Yes	~~Auto incremented~~	Yes	~~Surrogate~~ Primary Key	Number

				Positive integer			
CustomerName	First name and last name in full of customer	No	Yes	Any characters. Max 30 characters	No		Varchar2(30)
CustomerStatus	The particular status of a customer	good	Yes	Can either be: good or bad	No		Varchar2(5)
CustomerClassification	This is the type of customer a customer can classified as	No	Yes	Can either be: residential or non-residential	No		Varchar2(15)

Table: Service_Address

Attribute name	**Description**	**Any default values?**	**Required?**	**Allowable values**	**Unique?**	**Key?**	**Oracle data type/size**
AddressID	Unique Identifier	No	Yes	~~Auto incremented~~ Positive integer	Yes	~~Surrogate~~ Primary Key	Number
Address	The address of the service. Includes street number, street address, suburb and postcode	No	Yes	Any characters. Max 40 characters. Only includes street number, address, suburb and postcode. EG: 16 Hockley Loop, Canning Vale, 6155	No		Varchar2(40)
CustomerNo	The CustomerNo assigned to a particular address	No	No	As in customer table	No	Foreign Key REFERENCES Table: Customer (CustomerNo) ON delete Set null ON update cascade	Number

Table: Meter

Attribute name	Description	Any default values?	Required?	Allowable values	Unique?	Key?	Oracle data type/size
SerialNo	Unique Identifier	No	Yes	~~Auto incremented~~ Any characters. Max 15 characters or numbers	Yes	~~Surrogate~~ Primary Key	Varchar2(15)
DateInstalled	The date the Meter was installed in DD-MM-YYYY format	No	Yes	On or after 01-01-2019	No		Date
MeterStatus	The particular status of meter	working	Yes	Can either be: working or retired	No		Varchar2(7)
AddressID	The AddressID belonging to a particular meter	No	Yes	As in Service_Address table	No	Foreign Key REFERENCES Table: Service_Address (AddressID) ON delete cascade ON update cascade	Number

Table: Reader

Attribute name	Description	Any default values?	Required?	Allowable values	Unique?	Key?	Oracle data type/size
ReaderID	Unique Identifier	No	Yes	~~Auto incremented~~ Positive integer	Yes	~~Surrogate~~ Primary Key	Number
ReaderName	First name and last name in full of reader	No	Yes	Any characters. Max 30 characters	No		Varchar2(30)

Table: Reading

Attribute name	Description	Any default values	Required?	Allowable values	Unique?	Key?	Oracle data type/size
ReadingID	Unique Identifier	No	Yes	~~Auto incremented~~ Positive integer	Yes	~~Surrogate~~ Primary Key	Number
ReadingDate	Date of reading in DD-MM-YYYY format	No	Yes	On or after 01-01-2019	No		Date
ReadingAmount	The ReadingAmount of meter	No	Yes	Max 6-digit positive Integer	No		Number(6)
SerialNo	The ID of meter to be read	No	Yes	As in meter table	No	Foreign Key REFERENCES Table: Meter (SerialNo) ON delete no action ON update cascade	Varchar2(15)
ReaderID	The ID of reader doing the reading	No	Yes	As in reader table	No	Foreign Key REFERENCES Table: Reader (ReaderID) ON delete no action ON update cascade	Number

Table: Bill

Attribute name	Description	Any default values	Required?	Allowable values	Unique?	Key?	Oracle data type/size
BillNo	The bill number	No	Yes	Positive integer	No	Composite primary key	Number
ReadingID	The readingID belonging to a particular bill	No	Yes	As in reading table	No	Composite primary key Foreign Key REFERENCES Table: Reading	Number

						(ReadingID) ON delete no action ON update cascade	
PaymentMethod	The way customer paid the bill	No	No	Can either be: Paypal, Cash, or ETF	No		Varchar2(6)
BillingDate	Date of bill was created in DD-MM-YYYY format	No	Yes	On or after 01-01-2019	No		Date
DatePaid	Date bill is paid by the customer in DD-MM-YYYY format	No	No	On or after 01-01-2019	No		Date
CustomerNo	The CustomerNo the bill belongs to	No	Yes	As in customer table	No	Foreign Key REFERENCES Table: Customer (CustomerNo) ON delete no action ON update cascade	Number

Business rules:

- In Bill relation the PaymentMethod shouldn't be inserted if DatePaid hasn't been
- Assume new meter starts with initial ReadingAmount of 100,000 but it will not be recorded in database
- Assume 5 years of service means 5 year after installation (dateInstalled)
- Assume when CustomerStatus attribute is = 'bad' means the customer can't create a new account and they can't use their water supply

5a)

I created a trigger to check for the constraint. In addition, the trigger also creates a bill for the read as well. So, for every reading by a reader a bill is created immediately with that trigger as well. The trigger also does some simple error checking. For example, if a reader reads a meter on the 1/4/2020 then the date in which the meter was installed should be less than the data the meter was read. Essentially, a reader should not be able to read a meter that hasn't been installed yet

Code for constraint-

```
CREATE TRIGGER maxReadTrigger
       BEFORE INSERT
       ON Reading
    FOR EACH ROW

DECLARE
       numOfRead Int;
       numForBill Number;
       getAddress Number;
    getDateInstalled Date;
       numForCustomer Number;

BEGIN

       SELECT COUNT(EXTRACT(MONTH FROM ReadingDate)) Into numOfRead
       FROM Reading
       WHERE EXTRACT(MONTH FROM ReadingDate) =  EXTRACT(MONTH FROM
:New.ReadingDate)
       AND ReaderID = :New.ReaderID;

       Select MAX(BillNo) Into numForBill
       FROM Bill;

       Select AddressID Into getAddress
       FROM Meter
       WHERE SerialNo = :NEW.SerialNo;
    Select DateInstalled Into getDateInstalled
       FROM Meter
       WHERE SerialNo = :NEW.SerialNo;

    if getDateInstalled > :New.ReadingDate then
        raise_application_error(-20002, 'The reading date is not valid');
```

```
    End IF;


        Select CustomerNo into numForCustomer

        FROM Service_Address

        WHERE AddressID = getAddress;




        If numOfRead > 4 THEN

                raise_application_error(-20001, 'Reader has already read a max of 5
meters in a month');

        Else

                INSERT INTO Bill(BillNo, ReadingID, PaymentMethod, BillingDate,
DatePaid, CustomerNo)

                Values(numForBill+1, :New.ReadingID, null, :New.ReadingDate, null,
numForCustomer);

        End IF;


END;
```

5b)

For this I used a trigger. Every time a reader reads a meter for a customer they are checked as to whether they have paid their latest bill. If the customer has not paid within 60 days then their water supply will be cut off. This is represented by the customerStatus attribute in the customer relation being assigned 'bad'

```
select * from customer;
```

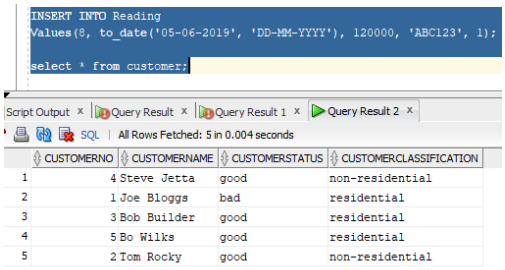
Query Result | All Rows Fetched: 5 in 0.003 seconds

	CUSTOMERNO	CUSTOMERNAME	CUSTOMERSTATUS	CUSTOMERCLASSIFICATION
1	4	Steve Jetta	good	non-residential
2	1	Joe Bloggs	good	residential
3	3	Bob Builder	good	residential
4	5	Bo Wilks	good	residential
5	2	Tom Rocky	good	non-residential

```
select * from bill;
```

Script Output | Query Result | All Rows Fetched: 7 in 0.004 seconds

	BILLNO	READ...	PAYMENTMETHOD	BILLINGDATE	DATEPAID	CUSTOMERNO
1	1	1	(null)	03/JAN/19	(null)	1
2	5	5	Paypal	20/NOV/20	21/NOV/20	2
3	6	6	Paypal	17/NOV/19	25/NOV/19	4
4	2	2	Cash	11/NOV/19	20/NOV/19	3
5	3	3	ETF	13/JUN/20	20/JUN/20	3
6	4	4	Paypal	23/JUL/20	02/AUG/20	3
7	7	7	Cash	28/AUG/20	29/AUG/20	3



```
INSERT INTO Reading
Values(8, to_date('05-06-2019', 'DD-MM-YYYY'), 120000, 'ABC123', 1);

select * from customer;
```

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

SQL | All Rows Fetched: 5 in 0.004 seconds

	CUSTOMERNO	CUSTOMERNAME	CUSTOMERSTATUS	CUSTOMERCLASSIFICATION
1	4	Steve Jetta	good	non-residential
2	1	Joe Bloggs	bad	residential
3	3	Bob Builder	good	residential
4	5	Bo Wilks	good	residential
5	2	Tom Rocky	good	non-residential

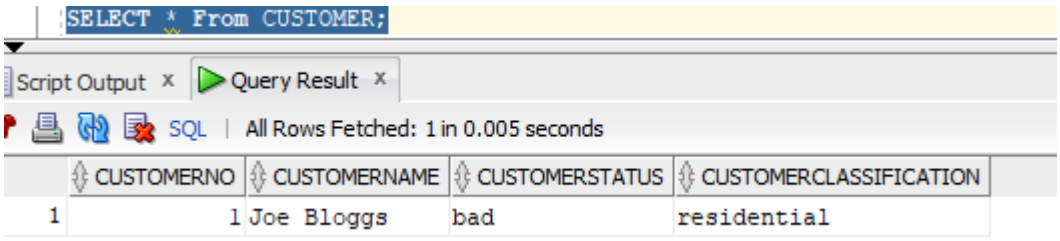
SQL for constraint 5b-

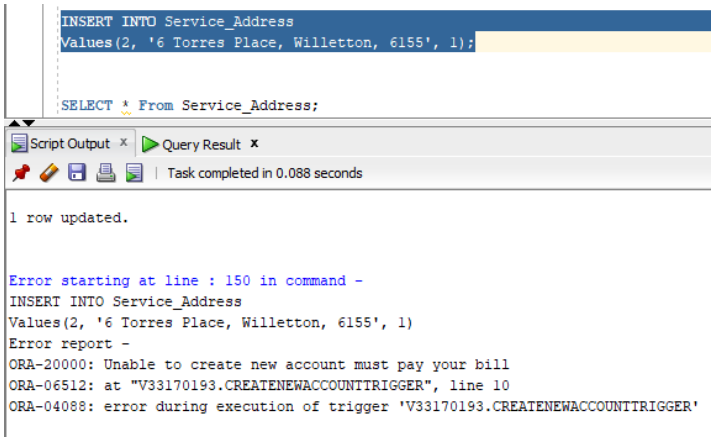
```
CREATE TRIGGER billNotPaidTrigger
        BEFORE INSERT
        ON Reading
    FOR EACH ROW

DECLARE
        getAddress Number;
        numForCustomer Number;
        numForBill Number;
        daysBeforeDue Int;
        dateBillDue DATE;

BEGIN
        SELECT AddressID into getAddress
        FROM METER
        WHERE SerialNo = :NEW.SerialNo;

        SELECT CustomerNo into numForCustomer
        FROM Service_Address
```

```
        WHERE AddressID = getAddress;

        SELECT BillNo Into numForBill
        FROM BILL
        WHERE CustomerNo = numForCustomer
        AND DatePaid IS NULL;

        If numForBill IS NOT NULL Then
                SELECT BillingDate + 60 Into dateBillDue
                FROM BILL
                WHERE BillNo = numForBill;
                daysBeforeDue := dateBillDue - :NEW.ReadingDate;
        End IF;

        If daysBeforeDue < 0  Then
                Update customer
                SET CustomerStatus = 'bad'
                WHERE CUSTOMERNO = numForCustomer;
        END IF;
END;
```

5c)

I used a trigger to implement this constraint. I utilised the customerStatus attribute to determine whether customer can create a new account. Below is the constraint in action





Code for constraint-

```
CREATE TRIGGER createNewAccountTrigger
	BEFORE INSERT
	ON Service_Address
	FOR EACH ROW

DECLARE
	status varchar2(5);
BEGIN

	SELECT CustomerStatus Into status
	FROM Customer
	WHERE CustomerNo = :new.CustomerNo;

	If status = 'bad' Then
		raise_application_error(-20000, 'Unable to create new account must pay your bill');
	End IF;

END;
```

6a)

The reason why I didn't need to insert in Bill is because my maxReadTrigger that I created for reading table automatically creates a bill on the same day. Thus, only updates are needed.

SQL statement for transaction-

```
INSERT INTO Customer
Values(3, 'Bob Builder', 'good', 'residential');

INSERT INTO Service_Address
Values(3, '53 Drip Drive, Dripwater, 1267', 3);

INSERT INTO Meter
Values('DDWC4763', to_date('30-10-2019', 'DD-MM-YYYY'), 'working', 3);

INSERT INTO Reading
Values(2, to_date('11-11-2019', 'DD-MM-YYYY'), 110250, 'DDWC4763', 1);

UPDATE BILL
SET PAYMENTMETHOD = 'Cash',
       DatePaid = to_date('20-11-2019', 'DD-MM-YYYY')
WHERE BillNo = 2;

COMMIT;
```

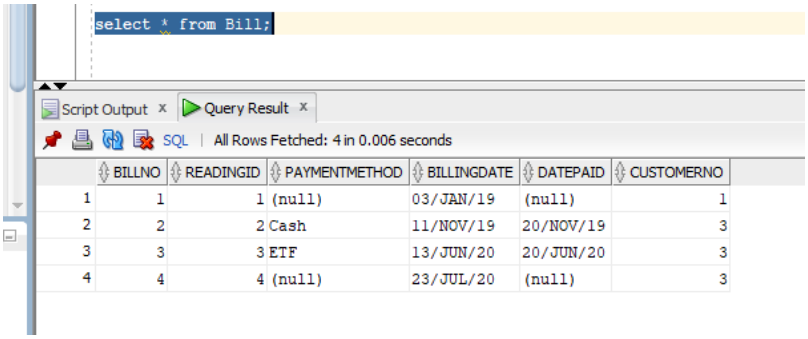
6b)

The trigger will find the customer that owns the water meter for 53 Drip Drive, Dripwater. In this case, the trigger (maxReadTrigger) identifies Bob Builder as being the owner. Bob Builder is customerNo 3

SQL statement for transaction-

```
INSERT INTO Reader
Values(2, 'Karen Karpenter');

INSERT INTO Reading
```

```
Values(3, to_date('13-06-2020', 'DD-MM-YYYY'), 123580, 'DDWC4763', 2);

UPDATE BILL
SET PAYMENTMETHOD = 'ETF',
       DatePaid = to_date('20-06-2020', 'DD-MM-YYYY')
WHERE BillNo = 3;


INSERT INTO Reading
Values(4, to_date('23-07-2020', 'DD-MM-YYYY'), 123683, 'DDWC4763', 2);

COMMIT;
```

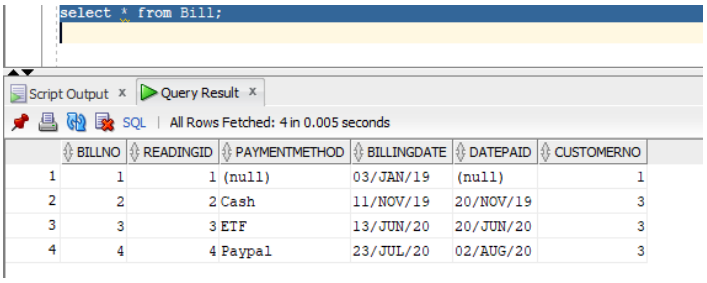


	BILLNO	READINGID	PAYMENTMETHOD	BILLINGDATE	DATEPAID	CUSTOMERNO
1	1	1	(null)	03/JAN/19	(null)	1
2	2	2	Cash	11/NOV/19	20/NOV/19	3
3	3	3	ETF	13/JUN/20	20/JUN/20	3
4	4	4	(null)	23/JUL/20	(null)	3

6c)

SQL statement for transaction-

```
UPDATE BILL
SET PAYMENTMETHOD = 'Paypal',
       DatePaid = to_date('02-08-2020', 'DD-MM-YYYY')
WHERE BillNo = 4;

COMMIT;
```



	BILLNO	READINGID	PAYMENTMETHOD	BILLINGDATE	DATEPAID	CUSTOMERNO
1	1	1	(null)	03/JAN/19	(null)	1
2	2	2	Cash	11/NOV/19	20/NOV/19	3
3	3	3	ETF	13/JUN/20	20/JUN/20	3
4	4	4	Paypal	23/JUL/20	02/AUG/20	3

7a)

SQL statement for View-

```
CREATE VIEW ViewA AS
SELECT READERID, COUNT(*) AS TotalReads
FROM Reading
WHERE to_char(readingDate, 'Month') = to_char(sysDate, 'Month')
GROUP BY READERID
WITH CHECK OPTION;
```

7b)

Assume the last reading in a new year is the first reading of the new year and not the last reading in the previous year

SQL for ViewB-

```
CREATE VIEW ViewB AS
select ReadingID, ReadingDate, READINGAMOUNT, readingamount - LAG(readingAmount, 1)
OVER (ORDER BY READINGAMOUNT) AS Consumption
from reading
WHERE EXTRACT(YEAR FROM ReadingDate) > 2019
AND SERIALNO = 'DDWC4763'
WITH CHECK OPTION;
```

7C)

Assumptions

- We are not dealing with leap years. So 6 months = 183 days (rounded up)
- Due to the fact I have a constraint that only allows meters to be installed on or after 01-01-2019, the meters will be replaced every year rather than every 5 years. This only applies to this view/section and does not apply system wide. This change is to illustrate that my view works and so you can see some sample data

SQL statement for View (applied to database)-

```
CREATE VIEW ViewC AS
Select SERIALNO
FROM METER
WHERE (Add_Months(DateInstalled, 12) - Current_Date) < 183
AND (Add_Months(DateInstalled, 12) - Current_Date) > 0
WITH CHECK OPTION;
```

SQL statement for View that I would have applied to database but can't due to the fact a meter can only be installed on or after 01-01-2019 (constraint). Thus 5 years would be 01-01-2025.

```
CREATE VIEW ViewC AS
Select SERIALNO
FROM METER
WHERE (Add_Months(DateInstalled, 60) - Current_Date) < 183
AND (Add_Months(DateInstalled, 60) - Current_Date) > 0
WITH CHECK OPTION;
```