INDEX		
S.NO	**CONTENT**	**NOS**
1	TABLES WITH VALUES	
2	SQL QUERY	12
3	VIEWS	8
4	PROCEDURE	4
5	FUNCTIONS	2
6	TRIGGERS	2
7	SEVERAL FUNCTIONS+TRIGGERS	2+1
8	ENTITY-RELATIONSHIP DIAGRAM	
9	RELATIONAL SCHEMA	

PROJECT TITLE: CAR WASHING SERVICE

Submitted for the course

CS7411- Database Management Systems Laboratory

NOTE:

1. HYPERLINKS ARE ATTACHED WITH EVERY TABLES, INDEX AND INDEX CONTENT. PLEASE USE IF NECESSARY
2. FOR EVERY ANSWER, A COLUMN “USEFULNESS” IS ADDED. IT PORTRAYS THE USEFULNESS OF THE RESPECTIVE QUERY/FUNCTION/PROEDURE/TRIGGER IN A REAL-TIME PROJECT.
3. DDL/DML COMMANDS ARE ALSO ATTACHED WITH THIS FILE. TO BE MORE CONVENIENT, WE CREATED TABLES WITH VALUES.
4. RELATIONAL SCHEMA AND ER DIAGRAM ARE ALSO AVAILABLE.

TABLES:

CUSTOMER: GO TO INDEX:

customer_id	customer_address	car_number	gender	customer_name	contact_no	office_id	customer_service_id
6001	Chennai	TN-06-A-6754	MALE	Robert	9882323012	2001	5001
6002	Chennai	TN-08-SR-7321	MALE	George	7902324221	2001	5002
6003	Chennai	TN-10-G-6032	FEMALE	Liz	6992813234	2001	5003
6004	Mumbai	MH-04-CN-8742	FEMALE	Mary	8902945482	2003	5004
6005	Mumbai	MH-03-CW-3464	MALE	Raj	7792013122	2003	5005
6006	Coimbatore	TN-10-AD-6809	MALE	Rohit	7987983128	2006	5006
6007	Madurai	TN-10-B-2325	MALE	Rajesh	8902038113	2008	5007
6008	Madurai	TN-07-BS-2734	MALE	John Kumar	9892238684	2008	5008
6009	Erode	TN-10-E-7892	MALE	Kumaran	9908261322	2010	5009
6010	Mumbai	MH-05-CM-7008	MALE	Lokesh	9878728243	2004	5010

CUSTOMER_SERVICE: GO TO INDEX:

customer_service_id	office_id	service_id	start_date	deliver_date
5001	2001	4001	2020-01-28	2020-01-29
5002	2001	4002	2020-01-29	2020-01-30
5003	2001	4003	2020-01-30	2020-01-30
5004	2003	4004	2020-02-01	2020-02-01
5005	2003	4005	2020-02-01	2020-02-02
5006	2006	4006	2020-01-24	2020-01-24
5007	2008	4007	2020-01-25	2020-01-25
5008	2008	4008	2020-01-26	2020-01-26
5009	2010	4009	2020-01-26	2020-01-27
5010	2004	4010	2020-01-27	2020-01-27

TOOKBY:

TOOKBY	
CUSTOMER_SERVICE_ID	**EMP_ID**
5001	3001
5002	3002
5003	3003
5004	3004
5005	3005
5006	3006
5007	3007
5008	3008
5009	3009
5010	3010

OFFICE:

OFFICE				
OFFICE_ID	**OFFICE_NAME**	**OFF_ADD_ST**	**OFF_ADD_DIS**	**OFF_ADD_STA**
2001	Car_care	45 Ranganathan st,west mambalam	Chennai	TamilNadu
2002	Diamond car wash	51 Saradapuram,mylapore	Chennai	TamilNadu
2003	splash washers	5 Andheri kurla road	Mumbai	Maharashtra
2004	majestic car wash	7 New link road,Andheri west	Mumbai	Maharashtra
2005	Car spa	699 avinashi road	Coimbatore	TamilNadu
2006	Mega car wash	41 Raja st	Coimbatore	TamilNadu
2007	Car shine	18 alanganallur	Madurai	TamilNadu
2008	Auto click	3 alagappa nagar	Madurai	TamilNadu
2009	Fast and clean	32 karungalpalayam	Erode	TamilNadu
2010	Royal Car wash	66 theppukulam	Erode	TamilNadu

SERVICE:

SERVICE				
SERVICE_ID	**DURATION**	**PAINT_COLOUR**	**TINKERING**	**WASH_ID**
4001	1 day	none	none	1001
4002	1 day	none	none	1001
4003	3 Hrs	none	none	1002
4004	6 Hrs	Red	Selected	1003
4005	1 day	Blue	Selected	1001
4006	4 Hrs	none	none	1004
4007	5 Hrs	Gray	none	1007
4008	8 Hrs	Red	Selected	1008
4009	1 day	none	Selected	1011
4010	6 Hrs	none	none	1005

BOOKING:

booking_id	pay_id	office_id	duration	service_id	customer_id	car_id
901	801	2001	1 day	4001	6001	701
902	802	2001	1 day	4002	6002	702
903	803	2001	3 Hrs	4003	6003	703
904	804	2003	6 Hrs	4004	6004	704
905	805	2003	1 day	4005	6005	705
906	806	2006	4 Hrs	4006	6006	706
907	807	2008	5 Hrs	4007	6007	707
908	808	2008	8 Hrs	4008	6008	708
909	809	2010	1 day	4009	6009	709
910	810	2004	6 Hrs	4010	6010	710

CAR:

colour	model	car_name	car_id	customer_id
Red	C-class	Benz	701	6001
Silver	Indica	TATA	702	6002
Blue	Verna	Hyundai	703	6003
White	X1	BMW	704	6004
Blue	Octavia	Skoda	705	6005
White	Swift	Suzuki	706	6006
Black	Corolla	Toyota	707	6007
Blue	800	Maruthi	708	6008
White	S90	Volvo	709	6009
Red	Linea	Fiat	710	6010

PAYMENT			
PAY_ID	TOTAL_PRICE	MODE_OF_PAYMENT	TRANS_STATUS
801	1000	cash	pending
802	1000	credit	pending
803	500	cash	success
804	500	cash	failed
805	1000	credit	success
806	400	cash	success
807	600	cash	pending
808	900	credit	pending
809	1100	credit	success
810	800	cash	failed

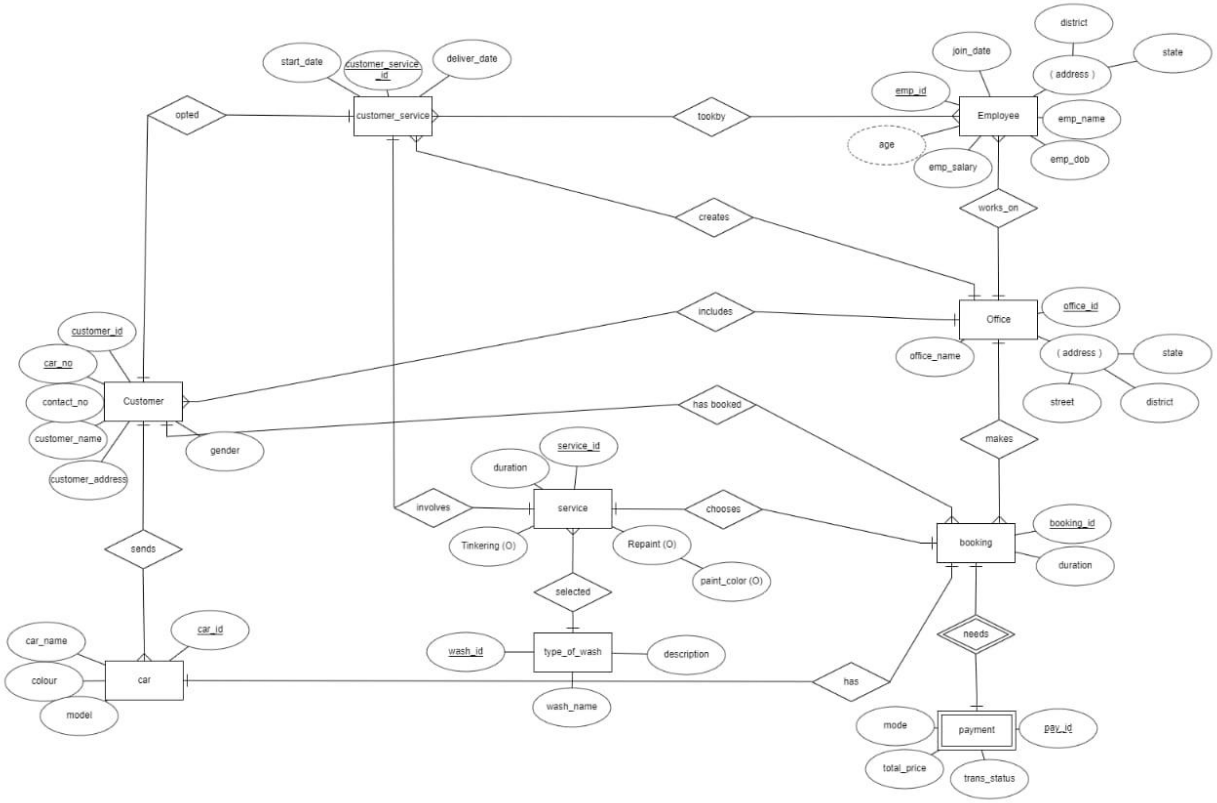
TYPE_OF_WASH		
WASH_ID	**WASH_NAME**	**DESCRIPTION**
1001	Handwash	minimizes scratching
1002	Waterless wash	uses spray bottles and microfiber towels
1003	Rinseless wash	hybrid of waterless and handwash
1004	Automatic wash	uses machinery -in conveyer belt
1005	Brushless wash	uses soft cloth - in machinery
1006	Touch less wash	uses pressure washers and pressurized air
1007	self-service wash	Allows customers to use car wash tools
1008	Auto-Detail wash	professional wash – interior and exterior wash
1009	steam car wash	Also kills bugs, germs eliminates wastewater
1010	pet washing	car wash including pet wash
1011	chemical wash	uses chemicals for bright look

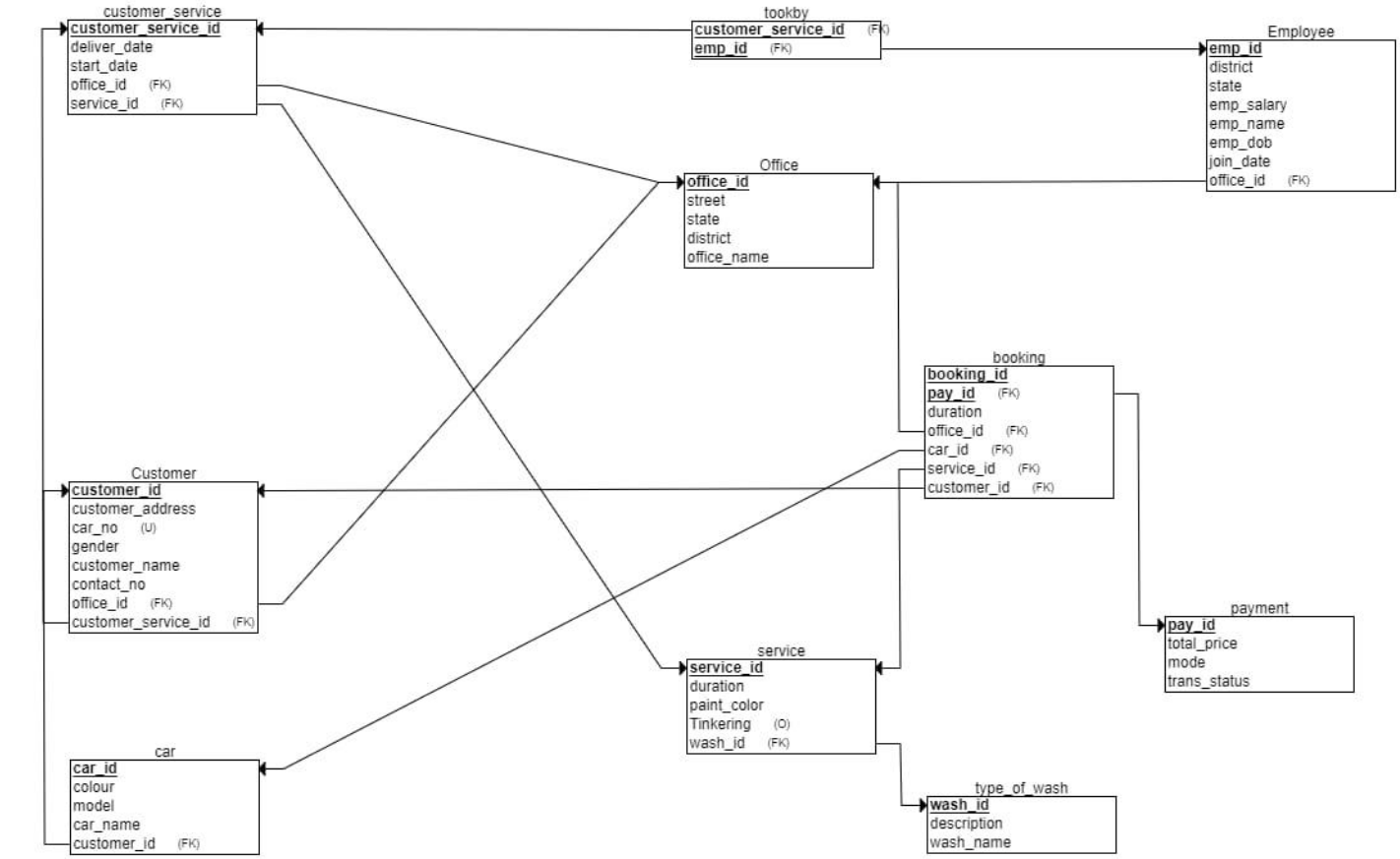
EMPLOYEE:

emp_id	emp_salary	emp_name	emp_dob	office_id	district	state	join_date
3001	7000	Michael	1990-01-24	2001	Chennai	TamilNadu	2010-08-17
3002	7000	Madhan	1993-04-02	2001	Chennai	TamilNadu	2012-09-24
3003	7500	kamaraj	1989-07-17	2001	Chennai	TamilNadu	2011-10-02
3004	9000	Dev	1995-03-06	2003	Mumbai	Maharashtra	2014-07-19
3005	9000	kumar	1992-09-10	2003	Mumbai	Maharashtra	2013-01-13
3006	7000	kathir	1992-01-21	2006	Coimbatore	TamilNadu	2013-03-16
3007	7000	Bala	1993-10-15	2008	Madurai	TamilNadu	2014-05-29
3008	8000	Adithya	1985-10-01	2008	Madurai	TamilNadu	2006-06-09
3009	8500	Patel	1985-12-12	2010	Erode	TamilNadu	2008-11-18
3010	8500	Guru	1987-06-19	2004	Mumbai	Maharashtra	2009-12-17

GO TO INDEX:

GO TO INDEX:


start_date
customer_service_id
deliver_date
customer_service
opted
tookby
emp_id
join_date
district
state
(address)
Employee
emp_name
age
emp_salary
emp_dob
works_on
creates
includes
Office
office_id
office_name
(address)
state
street
district
customer_id
car_no
contact_no
Customer
customer_name
customer_address
gender
has booked
makes
service_id
duration
involves
service
chooses
booking
booking_id
duration
Tinkering (O)
Repaint (O)
paint_color (O)
sends
selected
needs
car_name
car_id
colour
car
model
wash_id
type_of_wash
description
wash_name
has
mode
payment
pay_id
total_price
trans_status


customer_service
customer_service_id
deliver_date
start_date
office_id (FK)
service_id (FK)
tookby
customer_service_id (FK)
emp_id (FK)
Employee
emp_id
district
state
emp_salary
emp_name
emp_dob
join_date
office_id (FK)
Office
office_id
street
state
district
office_name
booking
booking_id
pay_id (FK)
duration
office_id (FK)
car_id (FK)
service_id (FK)
customer_id (FK)
Customer
customer_id
customer_address
car_no (U)
gender
customer_name
contact_no
office_id (FK)
customer_service_id (FK)
payment
pay_id
total_price
mode
trans_status
service
service_id
duration
paint_color
Tinkering (O)
wash_id (FK)
car
car_id
colour
model
car_name
customer_id (FK)
type_of_wash
wash_id
description
wash_name

SQL QUERY

GO TO INDEX:

DESCRIPTION(SQL QUERY):1

To find the Employee details who are joined in the office before any car's entry date and getting the 2nd highest salary in their office

QUERY:

```
select emp_name,emp_salary,to_char(join_date,'yyyy') as join_year from employee emp
inner join tookby took on took.emp_id=emp.emp_id
inner join customer_service cusser on cusser.customer_service_id=took.customer_service_id
where (abs(months_between(emp.join_date,cusser.start_date))>0)
and
emp_salary=(select min(emp_salary) from
(select distinct emp_salary from employee order by emp_salary desc)
where rownum<=2);
```

UESFULNESS:

Any employee who has joined the office before any car's arrival can be assigned to service that car. The employee with 2nd highest salary will seem to be an experienced one. By assigning him a higher level task, the output will be beneficial for the Administrator.

TABLE INVOLVED:

- tookby
- Customer_service
- Employee

CONSTRUCTS USED:

- Joins
- Subquery
- Date function
- Set operations

OUTPUT:

```
SQL> select emp_name,emp_salary,to_char(join_date,'yyyy') as join_year from empl
oyee emp inner join tookby took on took.emp_id=emp.emp_id inner join customer_se
rvice cusser on cusser.customer_service_id=took.customer_service_id where (abs(m
onths_between(emp.join_date,cusser.start_date))>0) and emp_salary=(select min(em
p_salary) from (select distinct emp_salary from employee order by emp_salary des
c)where rownum<=2);

EMP_NAME                                           EMP_SALARY JOIN
-------------------------------------------------- ---------- ----
Patel                                                    8500 2008
Guru                                                     8500 2009
```

DESCRIPTION(SQL QUERY):2

To find the full Employee details which is in the Nth(user's choice.Say for eg:5th row) of the table in the database.

QUERY:

```
select * from ( select a.*, rownum rnum from
 ( Select * from Employee e order by rowid asc ) a where rownum <= 5 ) where rnum >=5;
```

UESFULNESS:

Suppose any mistake/error made in list, we can fetch that row alone separately.

TABLE INVOLVED:

- Employee

CONSTRUCTS USED:

- Subquery
- Inbuilt-function

OUTPUT:

```
SQL> select * from ( select a.*, rownum rnum from ( Select * from Employee e ord
er by rowid asc ) a where rownum <= 5 ) where rnum >=5;

    EMP_ID EMP_SALARY EMP_NAME
---------- ---------- ------------------------------------------------
EMP_DOB    OFFICE_ID EMP_ADD_STA
--------- ---------- ------------------------------------------------
JOIN_DATE EMP_ADD_DIS                RNUM
--------- -------------------- ----------
      3005       9000 Kumar
10-OCT-92       2003 Maharashtra
13-JAN-13                               5
```

DESCRIPTION(SQL QUERY):3

To find the service which is selected mostly by the customers.

QUERY:

```
Select s.wash_id, count(s.wash_id),tow.wash_name,tow.description from Service s
inner join type_of_wash tow on tow.wash_id=s.wash_id
Group by s.wash_id,tow.wash_name,tow.description
Having count(s.wash_id)>=1
Order by count(s.wash_id) desc;
```

UESFULNESS:

It is very important in an administration to know about their service which is mostly used by the customers over a time. So that they can Brand their company with that service.(Say here: service named Handwash is mostly used by the customers)

TABLE INVOLVED:

- Employee

CONSTRUCTS USED:

- Subquery
- Inbuilt-function

OUTPUT:

```
   WASH_ID     MAX_WC WASH_NAME
---------- ---------- ------------------------------
DESCRIPTION
------------------------------------------------------------
      1001          3  Handwash
minimizes scratching

      1005          1  Brushless wash
uses soft cloth - in machinery

      1007          1  self-service wash
Allows customers to use car wash tools


   WASH_ID     MAX_WC WASH_NAME
---------- ---------- ------------------------------
DESCRIPTION
------------------------------------------------------------
      1008          1  Auto-Detail wash
professional wash - interior and exterior wash

      1002          1  Waterless wash
uses spray bottles and microfiber towels

      1003          1  Rinseless wash
hybrid of waterless and handwash


   WASH_ID     MAX_WC WASH_NAME
---------- ---------- ------------------------------
DESCRIPTION
------------------------------------------------------------
      1004          1  Automatic wash
uses machinery -in conveyer belt

      1011          1  chemical wash
uses chemicals for bright look


8 rows selected.
```

DESCRIPTION(SQL QUERY):4

To find the customer details who opted tinkering for their cars.(not in normal method.using string count)

QUERY:

```
select cus.customer_id,cus.customer_name,
REGEXP_COUNT ((substr(ser.tinkering,1,2)), 's', 1, 'i') as tinker
from booking book
inner join customer cus on cus.customer_id=book.customer_id
inner join customer_service cusser on cus.customer_service_id=cusser.customer_service_id
inner join service ser on ser.service_id=cusser.service_id ;
```

UESFULNESS:

In some tables, we have to use some attributes as radio button(selecting one value at a time). In such time, checking a string present or not, will be more useful.

TABLE INVOLVED:

- Booking
- Customer
- Customer_service
- Service

CONSTRUCTS USED:

- Subquery
- Inbuilt-function

OUTPUT:

```
SQL> select cus.customer_id,cus.customer_name,REGEXP_COUNT ((substr(ser.tinkerin
g,1,2)), 's', 1, 'i') as tinker from booking book inner join customer cus on cus
.customer_id=book.customer_id inner join customer_service cusser on cus.customer
_service_id=cusser.customer_service_id inner join service ser on ser.service_id=
cusser.service_id ;

CUSTOMER_ID CUSTOMER_NAME            TINKER
----------- -------------------- ----------
       6006 Rohit                        0
       6005 Raj                          0
       6004 Mary                         0
       6003 Liz                          0
       6002 George                       0
       6001 Robert                       0
       6007 Rajesh                       0
       6008 John kumar                   1
       6009 Kumaran                      1
       6010 Lokesh                       0

10 rows selected.
```

DESCRIPTION:5

To find all the customers, office, district whose transaction is ‘pending’ or ‘failed’.

QUERY:

```
select b.customer_id, o.office_id, o.district, p.total_price, p.trans_status
from booking b inner join office o on
b.office_id = o.office_id
inner join payment p on
b.pay_id = p.pay_id
where trans_status = 'pending'
or trans_status = 'failed';
```

USEFULNESS:

Suppose we want to change the transaction status which is failed or pending , this query would help to find those customers and thus save time.

TABLES INVOLVED:	CONSTRUCTS USED:
• booking • office • payment	• Inner Join

OUTPUT:

customer_id	office_id	district	total_price	trans_status
6001	2001	Chennai	1000	pending
6002	2001	Chennai	1000	pending
6004	2003	Mumbai	500	failed
6007	2008	Madurai	600	pending
6008	2008	Madurai	900	pending
6010	2004	Mumbai	700	failed

(6 rows)

DESCRIPTION:6

To find all the customers, office, transaction status which is booked on February month.

QUERY:

```
select b.customer_id, o.office_id, o.district, cs.deliver_date, p.trans_status from
booking b inner join customer_service cs on
b.service_id = cs.service_id
inner join office o on
b.office_id = o.office_id
inner join payment p on
b.pay_id = p.pay_id
where cs.start_date >= '01-FEB-2020'
and cs.deliver_date <= '28-FEB-2020';
```

USEFULNESS:

This query is used when we want information about all the transactions happened in a particular month (here : February) , thus helping in calculating company's profit or to find out whether there is any transaction which is pending or failed

TABLE INVOLVED:	CONSTRUCTS USED:
• booking • customer_service • office • payment	• Inner Join

OUTPUT:

```
customer_id | office_id | district | deliver_date | trans_status
------------+-----------+----------+--------------+--------------
       6004 |      2003 | Mumbai   | 2020-02-01   | failed
       6005 |      2003 | Mumbai   | 2020-02-02   | success
(2 rows)
```

DESCRIPTION:7

To find all the customers who has done payment in 'cash' in January month.

QUERY:

```
select c.customer_id,c.contact_no,p.mode,p.trans_status,
o.office_id,o.district
from customer c inner join booking b on
c.customer_id = b.customer_id
inner join payment p on
b.pay_id = p.pay_id
inner join customer_service cs on
b.service_id = cs.service_id
inner join office o on
o.office_id = b.office_id
where cs.start_date >= '01-JAN-2020'
and cs.deliver_date <= '31-JAN-2020'
and p.mode = 'Cash';
```

USEFULNESS:

This query will be useful in situations where a transaction has failed in a particular month (here: January) and want to contact that particular customer so as to make the transaction success.

TABLE INVOLVED:

- customer
- booking
- payment
- customer_service
- office

CONSTRUCTS USED:

- Inner Join

OUTPUT:

```
customer_id | contact_no | mode | trans_status | office_id |  district
------------+------------+------+--------------+-----------+-----------
       6003 | 6992813234 | Cash | success      |      2001 | Chennai
       6006 | 7987983128 | Cash | success      |      2006 | Coimbatore
       6007 | 8902038113 | Cash | pending      |      2008 | Madurai
       6010 | 9878728243 | Cash | failed       |      2004 | Mumbai
(4 rows)
```

DESCRIPTION:8

To find the office which has maximum no. of customers and display the full address of it

QUERY:

```
--create a view for count of office_id

 create view office_count as
 select office_id,count(office_id) as oc from booking
 group by office_id having count(office_id)>=1
 order by count(office_id) desc;

-- query

select o.office_id,o.office_name,n.oc,
concat(o.street, ',' ,o.district, ',' ,o.state) as office_address
from office o inner join office_count n on
o.office_id = n.office_id
where o.office_id = (select m.office_id from office_count m
where m.oc = (select max(m.oc) from office_count m));
```

USEFULNESS:

This query would help to find out the office branch and rise its employee's salary and also help to identify which office branch need not be focused to improve

TABLE INVOLVED:

- office
- office_count (view)

CONSTRUCTS USED:

- String function (concat)
- Inner Join

OUTPUT:

```
 office_id | office_name | oc |          office_address
-----------+-------------+----+-----------------------------------
      2001 | Car_care    |  3 | 45 Ranganthan st,Chennai,TamilNadu
(1 row)
```

DESCRIPTION:9

To find the details about customers' car and the employees worked, in the last week of January and also find how many days it has been, since it is delivered.

QUERY:

```
select c.customer_id,c.car_number,x.car_name,x.model,
b.office_id,e.emp_id,(current_date – cs.deliver_date) as days from
customer c inner join booking b on
c.customer_id = b.customer_id
inner join car x on
b.car_id = x.car_id
inner join customer_service cs on
b.service_id = cs.service_id
inner join tookby t on
t.customer_service_id = cs.customer_service_id
inner join employee e on
```

```
e.emp_id = t.emp_id
where e.emp_id < 3011 and
cs.start_date >= '25-JAN-2020' and
cs.deliver_date <= '31-JAN-2020' ;
```

USEFULNESS:

This query is useful to find information about a customers' car which was booked within a specified date and also to find out how frequently/regularly customers visit or which type of car models are brought frequently to car wash. – so as to make offers to those regular customers.

TABLE INVOLVED:

- booking
- customer
- car
- employee
- customer_service
- tookby

CONSTRUCTS USED:

- Inner Join
- Date function (current_date)

OUTPUT:

customer_id	car_number	car_name	model	office_id	emp_id	days
6001	TN-06-A-6754	Benz	C-class	2001	3001	61
6002	TN-08-SR-7321	TATA	Indica	2001	3002	60
6003	TN-10-G-6032	Hyundai	Verna	2001	3003	60
6007	TN-10-B-2325	Toyota	Corolla	2008	3007	65
6008	TN-07-BS-2734	Maruthi	800	2008	3008	64
6009	TN-10-E-7892	Volvo	S90	2010	3009	63
6010	MH-05-CM-7008	Fiat	Linea	2004	3010	63

(7 rows)

DESCRIPTION:10

To display those customers' who have opted for repaint and show the old colour and the new colour (to be painted) of the car.

QUERY:

```
select c.customer_id, c.car_number, x.colour as old , e.emp_id, e.district,
s.paint_colour  as new
from customer c inner join booking b on
c.customer_id = b.customer_id
inner join customer_service cs on
cs.service_id = b.service_id
inner join service s on
s.service_id = b.service_id
inner join tookby t on
cs.customer_service_id = t.customer_service_id
inner join employee e on
e.emp_id = t.emp_id
inner join car x on
x.car_id = b.car_id
where e.emp_id < 3011 and
```

s.paint_colour != 'none';

USEFULNESS:

This query is useful for employees to verify which cars are to be painted and what colour it should be painted. (to avoid confusion)

TABLE INVOLVED:

- booking
- customer
- service
- car
- employee
- customer_service
- tookby

CONSTRUCTS USED:

- Inner Join

OUTPUT:

customer_id	car_number	old	emp_id	district	new
6004	MH-04-CN-8742	White	3004	Mumbai	Red
6005	MH-03-CW-3464	Blue	3005	Mumbai	Blue
6007	TN-10-B-2325	Black	3007	Madurai	Gray
6008	TN-07-BS-2734	Blue	3008	Erode	Red

(4 rows)

QUESTION(SQL QUERY):11

To find the customer detail whose car wash can be finished and delivered at the same date of their booking and simultaneously check the customer whose order is opted for wash(ie., wash_id is allocated)

QUERY:

```
Select book.booking_id,cus.customer_id,cus.customer_name from booking book
inner join customer cus on cus.customer_id=book.customer_id
inner join customer_service cusser on
cus.customer_service_id=cusser.customer_service_id
where (cusser.deliver_date- cusser.start_date =0)
and
book.service_id in (
        (select service_id from booking)
        Intersect
        (select service_id from service where wash_id in (
                (select wash_id from service)
                union
                 (select wash_id from type_of_wash))
            )
);
```

UESFULNESS:

The most important to the overall success of your company is to utilize time in an efficient and productive manner.The output of this query helps the administrator to know about howmany customer's car have been quickly serviced by their company.

TABLE INVOLVED:	CONSTRUCTS USED:
• CUSTOMER • CUSTOMER_SERVICE • BOOKING	• Join • Subquery • Inbuilt functions(date function) • Set operations

OUTPUT:

```
SQL> Select book.booking_id,cus.customer_id,cus.customer_name from booking book
inner join customer cus on cus.customer_id=book.customer_id inner join customer_
service cusser on cus.customer_service_id=cusser.customer_service_id where (cuss
er.deliver_date- cusser.start_date =0) and book.service_id in ((select service_i
d from booking) intersect (select service_id from service where wash_id in ((sel
ect wash_id from service) union (select wash_id from type_of_wash))));

BOOKING_ID CUSTOMER_ID CUSTOMER_NAME
---------- ----------- --------------------
       903        6003 Liz
       904        6004 Mary
       906        6006 Rohit
       907        6007 Rajesh
       908        6008 John kumar
       910        6010 Lokesh
```

VIEWS

GO TO INDEX:

DESCRIPTION:

1. To create a view to display the customer name and the type of wash they have opted

NAME: opted_wash

QUERY:

```
CREATE OR REPLACE VIEW opted_wash AS
SELECT c.customer_id, c.customer_name, s.wash_id,  t.wash_name
FROM customer c
INNER JOIN booking b
ON c.customer_id = b.customer_id
INNER JOIN service s
ON b.service_id = s.service_id
INNER JOIN type_of_wash t
ON t.wash_id = s.wash_id;
```

USEFULNESS:

Used to get a simplified view of the type of wash of customers for the employees to work on.

TABLE INVOLVED:	CONSTRUCTS USED:
• customer • booking • service • type_of_wash	• Inner Join

OUTPUT:

customer_id	customer_name	wash_id	wash_name
6001	Robert	1001	Handwash
6002	George	1001	Handwash
6003	Liz	1001	Handwash
6004	Mary	1002	Waterless wash
6005	Raj	1001	Handwash
6006	Rohit	1004	Automatic wash
6007	Rajesh	1007	Self-service wash
6008	John kumar	1008	Auto-Detail wash
6009	Kumaran	1011	Chemical wash
6010	Lokesh	1005	Brushless wash

(10 rows)

DESCRIPTION:

2. To create a view to display the number of customers in each office branch.

NAME: office_count

QUERY:

```
CREATE OR REPLACE VIEW office_count AS
SELECT booking.office_id, count(booking.office_id) AS oc
FROM booking
GROUP BY booking.office_id
HAVING count(booking.office_id) >= 1
ORDER BY (count(booking.office_id)) DESC;
```

USEFULNESS:

Used to get to know which office branch needs more improvement and which doesn't.

TABLE INVOLVED:

- Booking

CONSTRUCTS USED:

- Group by statement

OUTPUT:

office_id	oc
2001	3
2003	2
2008	2
2006	1
2004	1
2010	1

(6 rows)

DESCRIPTION:

3. To create a view to display the number of employees working/worked on each customer service.

NAME: t1

QUERY:

```
CREATE OR REPLACE VIEW  t1  AS
SELECT tookby.customer_service_id, count(tookby.emp_id) AS emp_count
FROM tookby
GROUP BY tookby.customer_service_id
ORDER BY tookby.customer_service_id;
```

USEFULNESS:

Used to get to know how many number of employees are needed to work for each service based on duration.

TABLE INVOLVED:

- Tookby

CONSTRUCTS USED:

- Group by statement

OUTPUT:

```
 customer_service_id | emp_count
---------------------+-----------
                5001 |         1
                5002 |         1
                5003 |         1
                5004 |         1
                5005 |         1
                5006 |         1
                5007 |         1
                5008 |         1
                5009 |         1
                5010 |         1
(10 rows)
```

DESCRIPTION:

4. To create a view to display the number of employees working/worked on each customer service in TamilNadu.

NAME: tn_emp_count

QUERY:

```
CREATE OR REPLACE VIEW  tn_emp_count  AS
SELECT c.customer_id,  o.office_id, v.emp_count, b.duration,  o.district
FROM customer c
INNER JOIN t1 v ON c.customer_service_id = v.customer_service_id
INNER JOIN booking b ON c.customer_id = b.customer_id
INNER JOIN office o ON o.office_id = b.office_id
WHERE o.state = 'TamilNadu';
```

USEFULNESS:

Used to get to know how many number of employees are needed in TamilNadu to work for each service based on duration.

TABLE INVOLVED:

- customer
- booking

CONSTRUCTS USED:

- Inner Join

- office
- t1 (view)

OUTPUT:

customer_id	office_id	emp_count	duration	district
6001	2001	1	1 day	Chennai
6002	2001	1	1 day	Chennai
6003	2001	1	3 Hrs	Chennai
6006	2006	1	4 Hrs	Coimbatore
6007	2008	1	5 Hrs	Madurai
6008	2008	1	8 Hrs	Madurai
6009	2010	1	1 day	Erode

(7 rows)

DESCRIPTION:

5. To create a view to display the years of experience of each employee.

NAME: experience

QUERY:

```
CREATE OR REPLACE VIEW  experience  AS
SELECT employee.emp_id, (CURRENT_DATE - employee.join_date) / 365 AS exp
FROM employee
WHERE employee.emp_id < 3011;
```

USEFULNESS:

Used to allocate an employee with more years of experience for complicated services or to increase salary for those employees with more years of experience.

TABLE INVOLVED:

- Employee

CONSTRUCTS USED:

- date function (current_date)

OUTPUT:

emp_id	exp
3001	9
3002	7
3003	8
3004	5
3005	7
3010	10
3007	5
3009	11
3006	7
3008	13

(10 rows)

DESCRIPTION:6

To Create a view of the employee from the office in Chennai who is being paid more than Rs.8000

NAME: HIGH_PAY

QUERY:

```
CREATE OR replace view HIGH_PAY
AS
select distinct(emp.emp_name),emp.emp_salary from employee emp
inner join office offi on emp.office_id=offi.office_id
 where emp_salary>=8000
and
offi.office_add_dis='Chennai';
```

USEFULNESS:

Knowing every details of each employee is very important for the administration.

TABLE INVOLVED:

- Office
- Employee

CONSTRUCTS USED:

- Inner Join

OUTPUT:

```
EMP_NAME                                           EMP_SALARY
-------------------------------------------------- ----------
Adithya                                                  8000
```

DESCRIPTION:7

To Create a view of the customer whose allocated time for car washing is within 5 hours and has visited the office which is in their same district.

NAME: Quick_work

QUERY:

```
create view Quick_work
as
select
cus.customer_id,cus.customer_name,off.office_name,off.office_add_dis,cus.customer_address
from customer cus
inner join office off on off.office_id=cus.office_id
inner join booking book on cus.customer_id=book.customer_id
where cus.customer_address=off.office_add_dis
and
book.duration='5 Hrs';
```

USEFULNESS:

Administrator has to know about their customers.

TABLE INVOLVED:

- Customer
- office
- booking

CONSTRUCTS USED:

- Inner Join

OUTPUT:

```
CUSTOMER_ID CUSTOMER_NAME
----------- --------------------
OFFICE_NAME
--------------------------------------------------
OFFICE_ADD_DIS
--------------------------------------------------
CUSTOMER_ADDRESS
------------------------------------------------------------
       6007 Rajesh
Auto click
Madurai
Madurai
```

DESCRIPTION:8

To Create a view of the customer details who have done payment through cash.

NAME: Payment_by_cash

QUERY:

```
CREATE OR REPLACE VIEW Payment_by_cash AS
select c.customer_id,c.customer_name,b.duration,p.pay_id from
customer c inner join booking b on
c.customer_id = b.customer_id
inner join payment p on
```

p.pay_id = b.pay_id
where p.mode = 'Cash';

USEFULNESS:
In any business firm, Transaction details are very important and is also a crux to know about the mode of payment.

TABLE INVOLVED:

- Customer
- Booking
- Payment

CONSTRUCTS USED:

- Inner Join

OUTPUT:

customer_id	customer_name	duration	pay_id
6003	Liz	3 Hrs	803
6004	Mary	6 Hrs	804
6006	Rohit	4 Hrs	806
6007	Rajesh	5 Hrs	807
6010	Lokesh	6 Hrs	810

(5 rows)

PROCEDURE

GO TO INDEX

DESCRIPTION:

1. To change the wash id to 1002 of the customer id 6003 (service_id = 4003).

NAME: p_wash(int,int)

IN PARAMETERS:
c_id number
w_id number

PL\SQL COMMAND:

```
create or replace procedure p_wash( c_id number,w_id number)
is
begin
update service
set wash_id = w_id
where service_id = (select s.service_id from service s inner join booking b on
s.service_id = b.service_id
where b.customer_id = c_id);

exception
when others then
dbms_output.put_line( SQLERRM );
end;
```

```
--calling procedure

declare
c number := &c;                    --say here 6003
w number := &number;               --say here 1002
begin
p_wash(c,w);
end;
```

USEFULNESS:

Used to change the type of wash in case the customer wish to change even after booking.

TABLE INVOLVED:

- service
- booking

CONSTRUCTS USED:

- Inner Join
- Sub query
- Update

OUTPUT:

(before calling procedure – notice 4003 wash_id)

service_id	duration	paint_colour	tinkering	wash_id
4005	1 day	Blue	Selected	1001
4008	8 Hrs	Red	Selected	1008
4009	1 day	none	Selected	1011
4002	1 day	none	none	1001
4006	4 Hrs	none	none	1004
4007	5 Hrs	Gray	none	1007
4010	6 Hrs	none	none	1005
4004	6 hrs	Red	Selected	1002
4001	1 day	none	none	1001
4003	3 hrs	none	none	1001

(10 rows)

(after calling procedure – notice 4003 change in wash_id)

service_id	duration	paint_colour	tinkering	wash_id
4005	1 day	Blue	Selected	1001
4008	8 Hrs	Red	Selected	1008
4009	1 day	none	Selected	1011
4002	1 day	none	none	1001
4006	4 Hrs	none	none	1004
4007	5 Hrs	Gray	none	1007
4010	6 Hrs	none	none	1005
4004	6 hrs	Red	Selected	1002
4001	1 day	none	none	1001
4003	3 hrs	none	none	1002

(10 rows)

DESCRIPTION:

2. To change the status of transaction to ‘success’ of customer id 6001 (pay_id =

801).

NAME: status(int,varchar)

IN PARAMETERS:

c_id IN number
st IN varchar

PL\SQL COMMAND:

```
create or replace procedure status( c_id number,st varchar)
is
begin
update payment
set trans_status = st
where pay_id = (select p.pay_id from payment p inner join booking b on
p.pay_id = b.pay_id
where b.customer_id = c_id);

exception
when others then
dbms_output.put_line( SQLERRM );
end;


--calling procedure

declare
c number := &c;          --say here 6001
s varchar2 := &s;        --say here 'success'
begin
status(c,s);
end;
```

USEFULNESS:

Used to change the status of transaction of a particular customer id in case if the transaction is 'failed', 'pending', or 'success'.

TABLE INVOLVED:

- payment
- booking

CONSTRUCTS USED:

- Inner Join
- Sub query
- Update

OUTPUT:

(before calling procedure – notice 801 – trans_status)

pay_id	total_price	mode	trans_status
805	1000	Credit Card	success
804	500	Cash	failed
803	500	Cash	success
802	1000	Credit Card	pending
807	600	Cash	pending
808	900	Credit Card	pending
809	1100	Credit Card	success
806	500	Cash	success
810	700	Cash	failed
801	1000	cash	pending

(10 rows)

(after calling procedure – notice change in 801- trans_status)

pay_id	total_price	mode	trans_status
805	1000	Credit Card	success
804	500	Cash	failed
803	500	Cash	success
801	1000	cash	success
802	1000	Credit Card	pending
807	600	Cash	pending
808	900	Credit Card	pending
809	1100	Credit Card	success
806	500	Cash	success
810	700	Cash	failed

(10 rows)

DESCRIPTION:3

To change the contact no. of a customer in the customer table.

NAME: CHANGE_NO(NUMBER,IN OUT NUMBER)

IN PARAMETER:

C_ID NUMBER

IN AND OUT PARAMETER:

PH_NO IN OUT NUMBER

PL\SQL COMMAND:

```
SET SERVEROUTPUT ON SIZE 1000000;
CREATE or REPLACE PROCEDURE CHANGE_NO(C_ID NUMBER, PH_NO IN OUT
NUMBER)
IS
        v_ID NUMBER(4);
        VH_NO NUMBER(10);
        cursor CUR is
        select CUSTOMER_ID,CONTACT_NO from CUSTOMER where
CUSTOMER_ID=C_ID;
        id number;

BEGIN
        open CUR;
        loop
```

```
        fetch CUR into V_ID,VH_NO;
        EXIT WHEN CUR%NOTFOUND;
        end loop;
        close CUR;
        begin
        UPDATE CUSTOMER SET CONTACT_NO=PH_NO WHERE CUSTOMER_ID=C_ID;

         exception
         when others then
          dbms_output.put_line( SQLERRM );

end;

        END;
/

--calling procedure

DECLARE
        PHONE CUSTOMER.CONTACT_NO%type;
        no number;
BEGIN
        NO :=&NO;                       --say here 6001
        PHONE :=&PHONE;                 --say here 9790247864
        CHANGE_NO(NO,PHONE);
        dbms_output.put_line('YOUR NEW CONTACT NUMBER:'||PHONE||'IS
UPDATED');
END;
/
```

USEFULNESS:

Contact Number is an essential things in business for many purpose like marketing.
Contact Number is often changed by the users.
This procedure helps the update the new contact number from the customer.

TABLE INVOLVED:	**CONSTRUCTS USED:**
• Customer	• Update(DML)

OUTPUT:

(before calling procedure – notice 6001 customer_id)

```
SQL> select customer_id,contact_no from customer;

CUSTOMER_ID CONTACT_NO
----------- ----------
       6001 9882323012
       6002 7902324221
       6003 6992813234
       6004 8902945482
       6005 7792013122
       6006 7987983128
       6007 8902038113
       6008 9892238684
       6009 9908261322
       6010 9878728243
```

(after calling procedure – notice 6001's change in contact_no)

```
CUSTOMER_ID CONTACT_NO
----------- ----------
       6001 9790247864
       6002 7902324221
       6003 6992813234
       6004 8902945482
       6005 7792013122
       6006 7987983128
       6007 8902038113
       6008 9892238684
       6009 9908261322
       6010 9878728243
```

```
Enter value for no: 6001
old   5:        NO :=&NO;
new   5:        NO :=6001;
Enter value for phone: 9790247864
old   6:        PHONE :=&PHONE;
new   6:        PHONE :=9790247864;
YOUR NEW CONTACT NUMBER:9790247864IS UPDATED
```

DESCRIPTION:4

To change the mode of transaction for the given pay_id by the customer. But, Due to Noval corona virus disease, all the transactions are temporarily stopped.

NAME: CHANGE_PAYMENT(NUMBER,PAYMENT.MODE_OF_PAYMENT%TYPE)

IN PARAMETETS:

P_ID IN NUMBER

MODI IN PAYMENT.MODE_OF_PAYMENT%TYPE

PL\SQL COMMAND:

```
SET SERVEROUTPUT ON SIZE 1000000;
CREATE OR REPLACE PROCEDURE CHANGE_PAYMENT(P_ID IN NUMBER,MODI IN
PAYMENT.MODE_OF_PAYMENT%TYPE)
IS
CURSOR CUR IS
SELECT MODE_OF_PAYMENT FROM PAYMENT WHERE PAY_ID=P_ID;
OLD VARCHAR(20);
BEGIN
        open CUR;
        loop
        fetch CUR into OLD;
        EXIT WHEN CUR%NOTFOUND;
        end loop;
        close CUR;
        UPDATE PAYMENT  SET MODE_OF_PAYMENT=MODI WHERE PAY_ID=P_ID;
        dbms_output.put_line('WELCOME! DUE TO NOVEL CORONA DISEASE ATTACK,
ALL THE TRANSCATIONS ARE TEMPORAILY STOPPED.');
        dbms_output.put_line('YOUR OPTION IS NOT CHANGED.SORRY FOR
```

```
INCONVENIENCE!');
        UPDATE PAYMENT  SET MODE_OF_PAYMENT=OLD WHERE PAY_ID=P_ID;
         Exception
        when others then
         dbms_output.put_line( SQLERRM );


END;
/

--calling procedure

DECLARE
        NO number:=&NO;                         --say here 801
        modep varchar(10):=&modep;              --say here 'credit'

BEGIN

        CHANGE_PAYMENT(no,modep);
        --dbms_output.put_line('YOUR NEW PAY MODE'||PHONE||'IS UPDATED');
END;
/
```

USEFULNESS:

For the super rich, the virus is a market disruption—a **money** problem. For everyone else, it's **life** or death. Transactions are temporarily stopped. Stay safe and clean.

TABLE INVOLVED:

- payment

CONSTRUCTS USED:

- Sub query
- Update

OUTPUT:

(before calling procedure – notice 801 – mode_of_payment)

```
    PAY_ID MODE_OF_PA
---------- ----------
       801 cash
       802 credit
       803 cash
       804 cash
       805 credit
       806 cash
       807 cash
       808 credit
       809 credit
       810 cash
```

(after calling procedure – notice there is no change in 801- mode of payment)

```
WELCOME! DUE TO NOVEL CORONA DISEASE ATTACK, ALL THE TRANSCATIONS ARE
TEMPORARILY STOPPED.
YOUR OPTION IS NOT CHANGED.SORRY FOR INCONVENIENCE!

PL/SQL procedure successfully completed.
```

```
    PAY_ID MODE_OF_PA
---------- ----------
       801 cash
       802 credit
       803 cash
       804 cash
       805 credit
       806 cash
       807 cash
       808 credit
       809 credit
       810 cash
```

FUNCTIONS

GO TO INDEX:

DESCRIPTION:

1. To display the service opted by customer id 6002 (service id 4002).

NAME: get_service(c_id int)

PARAMETER:	**RETURN TYPE:**
num number := 6002;	SYS_REFCURSOR

PL\SQL COMMAND:

```
create or replace function get_service(c_id in number)
return sys_refcursor
as
cur sys_refcursor;
begin
open cur for
select c.customer_id,s.duration,s.paint_colour,s.tinkering,t.wash_name
from customer c inner join booking b
on c.customer_id = b.customer_id
inner join service s on
b.service_id = s.service_id
inner join type_of_wash t on
s.wash_id = t.wash_id

where c.customer_id = c_id;
return cur;
end;
/

--calling function

declare

cur sys_refcursor;
num number := 6002;
```

```
cur_id        customer.customer_id%type;
cur_dur       service.duration%type;
cur_paint     service.paint_colour%type;
cur_tin       service.tinkering%type;
cur_wash      type_of_wash.wash_name%type;

begin

cur := get_service(num);

loop
  fetch  cur  into  cur_id,cur_dur,cur_paint,cur_tin,cur_wash;
  exit when cur%notfound;

dbms_output.put_line(cur_id||'--'||cur_dur||'--'||cur_paint||'--'||cur_tin||'--'
||cur_wash);

end loop;

close cur;
end;
/
```

USEFULNESS:

Used to display the services opted by a particular customer, which gives clarity to the employees working on it. Thus, employees don't have to spend time on searching.

TABLE INVOLVED:

- customer
- booking
- service
- type_of_wash

CONSTRUCTS USED:

- Inner join
- Cursor & loop

OUTPUT:

(service table)

service_id	duration	paint_colour	tinkering	wash_id
4005	1 day	Blue	Selected	1001
4008	8 Hrs	Red	Selected	1008
4009	1 day	none	Selected	1011
4002	1 day	none	none	1001
4006	4 Hrs	none	none	1004
4007	5 Hrs	Gray	none	1007
4010	6 Hrs	none	none	1005
4004	6 hrs	Red	Selected	1002
4001	1 day	none	none	1001
4003	3 hrs	none	none	1002

(10 rows)

(after calling function – service_id = 4002 will printed)

```
            get_service
-----------------------------------
 6002--1 day--none--none--Handwash
(1 row)
```

DESCRIPTION:

2. To find whether employees 3006 , 3008 are eligible for salary rise (i.e. they should have at least 10 years of experience).

NAME: pro(e_id int)

PARAMETER:	**RETURN TYPE:**
num number := 3006;	sys_refcursor

PL\SQL COMMAND:

```
create or replace function pro(e_id in number)
return sys_refcursor
as
cur sys_refcursor;
begin
open cur for
select exp from experience where emp_id = e_id;
return cur;
end;
/

--calling function

declare

cur sys_refcursor;
num number := 3006;
ex number;

begin

cur := get_service(num);

loop
  fetch  cur  into  ex;
  exit when cur%notfound;

if  ex >= 10 then
dbms_output.put_line(ex||'--yrs of experience - eligible for salary rise');
else
dbms_output.put_line(ex||'--yrs of experience - not eligible for salary rise');
```

```
end if;
end loop;

close cur;
end;
/
```

USEFULNESS:
Used to find out which employees are eligible for salary rise and which are not by giving employee id.

TABLE INVOLVED:

- experience (view)

CO
US

OUTPUT:
(for emp_id = 3006)

```
                         pro
-----------------------------------------------------
 7--yrs of experience - not eligible for salary rise
(1 row)
```

(for emp_id = 3008)

```
                         pro
-----------------------------------------------------
 13--yrs of experience - eligible for salary rise
(1 row)
```

TRIGGERS

GO TO INDEX:

DESCRIPTION:

1. To store old payment and new payment implicitly total price is update for pay_id = 810

NAME: pay_change()

TRIGGER TYPE:
Before update

PL\SQL COMMAND:

```
-- create a table to store old price and new price

create table last_payment(
```

```
pay_id integer not null,
old_price integer not null,
mode varchar(20),
trans_status varchar(20),
new_price integer
);

-- create a function for trigger
CREATE FUNCTION pay_change()
RETURNS trigger
LANGUAGE 'plpgsql' AS
$BODY$
begin
if new.total_price <> old.total_price then
insert into last_payment (pay_id,old_price,mode,trans_status,new_price)
values(old.pay_id,old.total_price,old.mode,old.trans_status,new.total_price);
end if;
return new;
end;
$BODY$;

-- creating a trigger

create trigger upd_price
before update
on payment
for each row
execute procedure pay_change();
```

USEFULNESS:
Used to store the transaction history of customers who have attempted to update the total_price, thus maintaining a fair transaction.

TABLE INVOLVED:

- payment
- last_payment (newly created)

CONSTRUCTS USED:

- Comparison Operators
- Control Structures
- DML (insertion in function)

OUTPUT:
select * from payment;

pay_id	total_price	mode	trans_status
801	1000	cash	success
802	1000	Credit Card	pending
803	500	Cash	success
804	500	Cash	failed
805	1000	Credit Card	success
806	500	Cash	success
807	600	Cash	pending
808	900	Credit Card	pending
809	700	Credit Card	success
810	800	Cash	failed

(10 rows)

update payment
set total_price = 700
where pay_id = 810;

select * from last_payment;

pay_id	old_price	mode	trans_status	new_price
810	800	Cash	failed	700

(1 row)

DESCRIPTION:

2. To add 100 Rs. to total_price in payment table implicitly when tinkering is 'Selected' in service table where service_id = 4003

NAME: tk()

TRIGGER TYPE:
after update

PL\SQL COMMAND:

```
CREATE FUNCTION tk()
RETURNS trigger
LANGUAGE 'plpgsql' AS
$BODY$
begin
if new.tinkering = 'Selected' then
update payment
set total_price = total_price + 100
where pay_id = (select pay_id from booking where
booking.service_id = old.service_id);
end if;
return new;
end;
$BODY$;

-- create trigger
```

```
create trigger tg_price
after update
on service
for each row
execute procedure tk();
```

USEFULNESS:

(Tinkering = 100 Rs.) In case the customer who didn't opt for tinkering during booking may select tinkering after sometime. This function hence, add 100 Rs. to total_price in payment table implicitly while we update tinkering to 'Selected'. Also this is reflected in last_payment table. Thus saving search time.

TABLE INVOLVED:

- payment
- booking
- service

CONSTRUCTS USED:

- Control Structures
- Arithmetic Operators
- Sub queries

OUTPUT:

select * from service;

service_id	duration	paint_colour	tinkering	wash_id
4005	1 day	Blue	Selected	1001
4008	8 Hrs	Red	Selected	1008
4009	1 day	none	Selected	1011
4002	1 day	none	none	1001
4006	4 Hrs	none	none	1004
4007	5 Hrs	Gray	none	1007
4010	6 Hrs	none	none	1005
4004	6 hrs	Red	Selected	1002
4001	1 day	none	none	1001
4003	3 hrs	none	none	1002

(10 rows)

select * from payment;

(notice 803 : total_price = 500)

pay_id	total_price	mode	trans_status
801	1000	cash	success
802	1000	Credit Card	pending
803	500	Cash	success
804	500	Cash	failed
805	1000	Credit Card	success
806	500	Cash	success
807	600	Cash	pending
808	900	Credit Card	pending
809	700	Credit Card	success
810	800	Cash	failed

(10 rows)

update service
set tinkering = 'Selected'
where service = 4003;

select * from payment;
(notice 803 : total_price = 600)

pay_id	total_price	mode	trans_status
801	1000	cash	success
802	1000	Credit Card	pending
804	500	Cash	failed
805	1000	Credit Card	success
806	500	Cash	success
807	600	Cash	pending
808	900	Credit Card	pending
809	700	Credit Card	success
810	700	Cash	failed
803	600	Cash	success

(10 rows)

select * from last_payment;

pay_id	old_price	mode	trans_status	new_price
810	800	Cash	failed	700
803	500	Cash	success	600

(2 rows)

GO TO INDEX:

Now we demonstrate a several functions executing consequently and respective triggers are called.

PROCESS:

STEP1:

For a given customer_id, check their transaction_status and return the value(pending/success/failed) → we used a function **check_transaction(number)**

STEP2:

If transaction is 'pending', inform as 'Your Transaction is in pending'

If transaction is 'success', inform as 'Your Transaction is success'

If transaction is ‘failed’, inform as ‘Your Transaction is in pending’ → we used a function **alloc_ser(varchar)**

STEP3:

If the transaction is failed, change their car washing duration to ‘0’.

STEP4:

If any change occurred in duration, a trigger will display the old and new value of duration. → we used a trigger **display_ID_changes**

FUNCTION NAME: check_transaction(number)
To check the transaction whether it is completed or not for the given customer

```
set serveroutput on size 100000;

create or replace function check_transaction(cus_id in number)
return sys_refcursor
as
check_cur sys_refcursor;
begin
open check_cur for
select cus.customer_id, cusser.customer_service_id,pay.trans_status from
customer cus inner join customer_service cusser on
cus.customer_service_id=cusser.customer_service_id inner join booking book on
cus.customer_id=book.customer_id inner join payment pay on
book.pay_id=pay.pay_id where cus.customer_id=cus_id;
return check_cur;
end;
/
```

CALLING THE FUNCTIONS
Giving input to functions

```
declare

check1_cur sys_refcursor;

no number := &no;              -- SAY HERE 6010

d_cus_id customer.customer_id%type;
d_trans_status payment.trans_status%type;
detail varchar2( 50 ):= '';
d_cus_ser_id customer_service.customer_service_id%type;
```

```
begin

check1_cur := check_transaction(no);

loop

fetch check1_cur into d_cus_id,d_cus_ser_id, d_trans_status;
exit when check1_cur%notfound;

dbms_output.put_line('CUSTOMERE ID :'||d_cus_id||'------>'||'TRANSACTION
STATUS:' || d_trans_status);
end loop;
close check1_cur;
detail := alloc_ser(d_trans_status);
dbms_output.put_line(detail);
if detail = 'Your Transaction is failed' then
BEGIN
  UPDATE booking
  SET duration ='0' WHERE booking_ID=(select book.booking_id from customer
cus inner join booking book on cus.customer_id=book.customer_id where
cus.customer_id=no) ;

END;

end if;

end;
/
```

TRIGGER NAME: display_ID_changes

When any changes in table booking occurs trigger is fired

```
set serveroutput on size 100000;

CREATE OR REPLACE TRIGGER display_ID_changes
BEFORE DELETE OR INSERT OR UPDATE ON booking
FOR EACH ROW
DECLARE
new_val varchar(50);
BEGIN
  dbms_output.put_line('Estimated time to wash your car:' || :OLD.duration);
  dbms_output.put_line('Your Transaction is failed. So time is set to' ||
:NEW.duration);
  dbms_output.put_line('Please Do the Transaction Again! Thank You');
END;
```

/

GO TO INDEX:

OUTPUT:

```
Enter value for no: 6010
old   5: no number := &no;
new   5: no number := 6010;
CUSTOMERE ID :6010------>TRANSACTION STATUS:failed
Your Transaction is failed
Estimated time to wash your car:8 Hrs
Your Transaction is failed. So time is set to0
Please Do the Transaction Again! Thank You

PL/SQL procedure successfully completed.
```

Trigger