



TIS3351 - ADVANCED DATABASE

Assignment 2

Data Warehouse for GrabCar

SUBMISSION DATE: 26/09/2019

PREPARED BY:

NAME	ID	EMAIL
Baobaid, Husam Omar Ali	1161304070	1161304070@student.mmu.edu.my
Muhammad Wafi bin Othman	1161304576	1161304576@student.mmu.edu.my
Mansoor	1161303225	1161303225@student.mmu.edu.my
Mohab Mohamed	1161302776	1161302776@student.mmu.edu.my

TABLE OF CONTENTS

1.Procedural SQL	3
a. Trigger	4
b. Stored Procedure	6
c. User-defined function	8
2. SQL	10
a. Joins	10
b. Group by/Group by Rollup/Group by Cube and having clause	11
c. 1 SQL each on the following operator:	12
• Comparison operators: =, <, <=, >=, <>	12
• Logical operator OR, AND, NOT	13
• Special operator BETWEEN, IS NULL, LIKE, IN, EXISTS	14
• Aggregate operator COUNT, MAX, MIN, SUM, AVG	15
d. View	16
e. TWO SQL not covered in lecture	17

1.Procedural SQL

DML	NAME	DESCRIPTION
Trigger	trgTripPrice	Sets the Trip_Price based on the hours the customer spent during trip. Price is also considered by the type of the vehicle. Any promotion codes applied is also considered.
Stored Procedure	addPoints	Adds the points earned by the customer from the trip. Points are calculated based on the trip price where 1x point <= 10 Ringgit, 1.5x point 11-20 Ringgit, 2x point 21-39 Ringgit, 3x point >= 40 Ringgit.
User-defined function	CusPoint(CID CHAR(4))	Returns a table of Reward_ID and total points of the customer(Sums all points earned from trips and added with previous points from the account. CID == Reward_ID

a. Trigger

Trips table before trigger:

```
SELECT * FROM trips;
```

TRIP_ID	CUS_ID	DRIVER_ID	VEH_ID	REWARD_ID	TIME_ID	LOC_ID	PROMO_ID	POINTS_EARNED	TRIP_PRICE
TR0002	C0002	DR05	V005	R002	T002	L010	20195	0	0.00
TR0003	C0003	DR07	V007	R003	T003	L008	20194	0	0.00
TR0004	C0004	DR01	V001	R004	T004	L011	20183	0	0.00
TR0005	C0005	DR06	V006	R005	T005	L001	20179	0	0.00
TR0006	C0006	DR01	V010	R006	T006	L002	20191	0	0.00
TR0007	C0007	DR11	V011	R007	T007	L012	20192	0	0.00
TR0008	C0008	DR17	V017	R008	T008	L013	21181	0	0.00
TR0009	C0009	DR19	V019	R009	T009	L002	20191	0	0.00
TR0010	C0010	DR19	V019	R010	T010	L007	21181	0	0.00
TR0011	C0001	DR15	V015	R001	T011	L006	20177	0	0.00
TR0012	C0014	DR14	V014	R014	T012	L010	20185	0	0.00
TR0013	C0018	DR03	V003	R018	T013	L020	20196	0	0.00
TR0014	C0020	DR02	V002	R020	T014	L019	20195	0	0.00
TR0015	C0019	DR01	V001	R019	T015	L005	21180	0	0.00
TR0016	C0002	DR08	V008	R002	T016	L015	20179	0	0.00
TR0017	C0016	DR19	V019	R016	T017	L009	20184	0	0.00
TR0018	C0011	DR09	V009	R011	T018	L004	20194	0	0.00
TR0019	C0015	DR16	V016	R015	T019	L003	20195	0	0.00
TR0020	C0007	DR13	V013	R007	T020	L012	20188	0	0.00
TR0001	C0001	DR01	V001	R001	T001	L005	20197	0	0.00

```
CREATE TRIGGER trgtripprice
AFTER INSERT ON trips
FOR EACH ROW MODE DB2SQL
UPDATE trips
SET trip_price =
(SELECT veh_price_hour FROM vehicle
WHERE trips.veh_id = vehicle.veh_id)*
(SELECT CASE Timestampdiff(8,CHAR(arrival_time-booking_time))
  WHEN 0 THEN 1
  ELSE Timestampdiff(8,CHAR(arrival_time-booking_time)) + 1
END
FROM TIME
WHERE trips.time_id = TIME.time_id)
- (SELECT promo_amount FROM promotion
  WHERE promotion.promo_id = trips.promo_id);
```

Trips table after trigger:

```
SELECT * FROM trips;
```

TRIP_ID	CUS_ID	DRIVER_ID	VEH_ID	REWARD_ID	TIME_ID	LOC_ID	PROMO_ID	POINTS_EARNED	TRIP_PRICE
TR0003	C0003	DR07	V007	R003	T003	L008	20194	0	17.78
TR0004	C0004	DR01	V001	R004	T004	L011	20183	0	18.50
TR0005	C0005	DR06	V006	R005	T005	L001	20179	0	116.70
TR0006	C0006	DR01	V010	R006	T006	L002	20191	0	7.00
TR0007	C0007	DR11	V011	R007	T007	L012	20192	0	29.30
TR0008	C0008	DR17	V017	R008	T008	L013	21181	0	17.40
TR0009	C0009	DR19	V019	R009	T009	L002	20191	0	24.00
TR0010	C0010	DR19	V019	R010	T010	L007	21181	0	65.40
TR0011	C0001	DR15	V015	R001	T011	L006	20177	0	18.00
TR0012	C0014	DR14	V014	R014	T012	L010	20185	0	36.90
TR0013	C0018	DR03	V003	R018	T013	L020	20196	0	7.99
TR0014	C0020	DR02	V002	R020	T014	L019	20195	0	15.40
TR0015	C0019	DR01	V001	R019	T015	L005	21180	0	5.50
TR0016	C0002	DR08	V008	R002	T016	L015	20179	0	17.70
TR0017	C0016	DR19	V019	R016	T017	L009	20184	0	12.00
TR0018	C0011	DR09	V009	R011	T018	L004	20194	0	47.78
TR0019	C0015	DR16	V016	R015	T019	L003	20195	0	15.40
TR0020	C0007	DR13	V013	R007	T020	L012	20188	0	9.00
TR0001	C0001	DR01	V001	R001	T001	L005	20197	0	8.60
TR0002	C0002	DR05	V005	R002	T002	L010	20195	0	28.40

b. Stored Procedure

Reward table before Stored Procedure:

```
SELECT * FROM trips;
```

TRIP_ID	CUS_ID	DRIVER_ID	VEH_ID	REWARD_ID	TIME_ID	LOC_ID	PROMO_ID	POINTS_EARNED	TRIP_PRICE
TR0003	C0003	DR07	V007	R003	T003	L008	20194	0	17.78
TR0004	C0004	DR01	V001	R004	T004	L011	20183	0	18.50
TR0005	C0005	DR06	V006	R005	T005	L001	20179	0	116.70
TR0006	C0006	DR01	V010	R006	T006	L002	20191	0	7.00
TR0007	C0007	DR11	V011	R007	T007	L012	20192	0	29.30
TR0008	C0008	DR17	V017	R008	T008	L013	21181	0	17.40
TR0009	C0009	DR19	V019	R009	T009	L002	20191	0	24.00
TR0010	C0010	DR19	V019	R010	T010	L007	21181	0	65.40
TR0011	C0001	DR15	V015	R001	T011	L006	20177	0	18.00
TR0012	C0014	DR14	V014	R014	T012	L010	20185	0	36.90
TR0013	C0018	DR03	V003	R018	T013	L020	20196	0	7.99
TR0014	C0020	DR02	V002	R020	T014	L019	20195	0	15.40
TR0015	C0019	DR01	V001	R019	T015	L005	21180	0	5.50
TR0016	C0002	DR08	V008	R002	T016	L015	20179	0	17.70
TR0017	C0016	DR19	V019	R016	T017	L009	20184	0	12.00
TR0018	C0011	DR09	V009	R011	T018	L004	20194	0	47.78
TR0019	C0015	DR16	V016	R015	T019	L003	20195	0	15.40
TR0020	C0007	DR13	V013	R007	T020	L012	20188	0	9.00
TR0001	C0001	DR01	V001	R001	T001	L005	20197	0	8.60
TR0002	C0002	DR05	V005	R002	T002	L010	20195	0	28.40

```
CREATE PROCEDURE addpoints()
UPDATE trips
SET     points_earned = CASE
        WHEN trip_price <= 10 THEN trip_price
        WHEN trip_price BETWEEN 11 AND 20 THEN
            trip_price * 1.5
        WHEN trip_price BETWEEN 21 AND 39 THEN trip_price * 2
        WHEN trip_price >= 40 THEN trip_price * 3
    end;
```

Reward table after Stored Procedure:

call addPoints();

```
SELECT * FROM trips;
```

TRIP_ID	CUS_ID	DRIVER_ID	VEH_ID	REWARD_ID	TIME_ID	LOC_ID	PROMO_ID	POINTS_EARNED	TRIP_PRICE
TR0003	C0003	DR07	V007	R003	T003	L008	20194	26	17.78
TR0004	C0004	DR01	V001	R004	T004	L011	20183	27	18.50
TR0005	C0005	DR06	V006	R005	T005	L001	20179	350	116.70
TR0006	C0006	DR01	V010	R006	T006	L002	20191	7	7.00
TR0007	C0007	DR11	V011	R007	T007	L012	20192	58	29.30
TR0008	C0008	DR17	V017	R008	T008	L013	21181	26	17.40
TR0009	C0009	DR19	V019	R009	T009	L002	20191	48	24.00
TR0010	C0010	DR19	V019	R010	T010	L007	21181	196	65.40
TR0011	C0001	DR15	V015	R001	T011	L006	20177	27	18.00
TR0012	C0014	DR14	V014	R014	T012	L010	20185	73	36.90
TR0013	C0018	DR03	V003	R018	T013	L020	20196	7	7.99
TR0014	C0020	DR02	V002	R020	T014	L019	20195	23	15.40
TR0015	C0019	DR01	V001	R019	T015	L005	21180	5	5.50
TR0016	C0002	DR08	V008	R002	T016	L015	20179	26	17.70
TR0017	C0016	DR19	V019	R016	T017	L009	20184	18	12.00
TR0018	C0011	DR09	V009	R011	T018	L004	20194	143	47.78
TR0019	C0015	DR16	V016	R015	T019	L003	20195	23	15.40
TR0020	C0007	DR13	V013	R007	T020	L012	20188	9	9.00
TR0001	C0001	DR01	V001	R001	T001	L005	20197	8	8.60
TR0002	C0002	DR05	V005	R002	T002	L010	20195	56	28.40

c. User-defined function

Before calling the function:

```
SELECT reward_id, no_of_points
FROM reward
WHERE reward_id = 'R001' OR
reward_id = 'R002';
```

REWARD_ID	NO_OF_POINTS
R001	10000
R002	1300

```
SELECT reward_id , points_earned
FROM trips
WHERE reward_id = 'R001'OR
reward_id = 'R002';
```

REWARD_ID	POINTS_EARNED
R001	27
R002	26
R001	8
R002	56

```
CREATE FUNCTION cuspoint(cid CHAR(4))
RETURNS TABLE ( rewardno CHAR(4), pointsavailable INT )
LANGUAGE SQL
RETURN (SELECT reward.reward_id,
              (SELECT DISTINCT( reward.no_of_points
                               + (SELECT Sum(trips.points_earned)AS NEW
                                   FROM   trips
                                   WHERE  trips.reward_id = cid) ) AS
              PointsAvailable
              FROM   reward,
                    trips
              WHERE  reward.reward_id = trips.reward_id
                    AND reward.reward_id = cid)
FROM   trips,
      reward
WHERE  reward.reward_id = trips.reward_id
      AND reward.reward_id = cid
GROUP BY reward.reward_id);
```



```
SELECT * FROM Table(Cuspoint('R001'));
```

REWARDNO	POINTSAVAILABLE
R001	10035

```
SELECT * FROM Table(Cuspoint('R002'));
```

REWARDNO	POINTSAVAILABLE
R002	1382

2. SQL

a. Joins

To join 3 tables, trips, location, and time. To see the frequent trips in certain place on certain time of the day so the grab can send more driver at certain place at specific time

```
SELECT location.loc_desc,  
       TIME.time_clocktime,  
       trips.trip_price  
FROM   (( trips  
        INNER JOIN location  
          ON trips.loc_id = location.loc_id)  
        INNER JOIN TIME  
          ON trips.time_id = TIME.time_id)  
ORDER BY TIME.time_clocktime asc;
```

LOC_DESC	TIME_CLOCKTIME	TRIP_PRICE
IOI City Mall	12:50:00 AM	18.50
CIMB Bank	4:00:00 AM	28.40
Pavilion Mall	4:02:00 AM	65.40
Parklane OUG	6:35:00 AM	24.00
Serin residency	7:20:00 AM	116.70
Merchantrade	8:05:00 AM	17.70
Parklane OUG	9:13:00 AM	7.00
Sunway Pyramid	9:17:00 AM	12.00
Tune hotel	9:45:00 AM	9.00
MMU	10:00:00 AM	47.78
KLCC	10:45:00 AM	15.40
KLIA	11:04:00 AM	5.50
Masjid Sultan Salahud...	12:00:00 PM	18.00
Tune hotel	1:07:00 PM	29.30
Ministry of Education	2:28:00 PM	15.40
KLIA	2:45:00 PM	8.60
Gurney Plaza	5:15:00 PM	17.78
The Mines	6:11:00 PM	17.40
CIMB Bank	9:30:00 PM	36.90
KPJ Hospital	11:55:00 PM	7.99

b. Group by/Group by Rollup/Group by Cube and having clause

Lists number of trips between 2018/19 by 3 facts vehicle, location and time

```

SELECT veh_type      AS VehicleType,
       loc_desc      AS Location,
       time_year     AS Year,
       Count(trip_id) AS NumOfTrips
FROM   vehicle,
       location,
       time,
       trips
WHERE  vehicle.veh_id = trips.veh_id
      AND trips.loc_id = location.loc_id
      AND trips.time_id = time.time_id
GROUP BY veh_type,
          cube( loc_desc, time_year )
HAVING time_year >= 2018
ORDER BY vehicletype,
         location,
         year

```

VEHICLETYPE	LOCATION	YEAR	NUMOFTRIPS
MPV	KLCC	2019	1
MPV	Ministry of Education	2019	1
MPV	Parklane OUG	2019	2
MPV	Pavilion Mall	2018	1
MPV	Sunway Pyramid	2018	1
MPV	Tune hotel	2019	1
MPV		2018	2
MPV		2019	5
SEDAN	CIMB Bank	2019	1
SEDAN	IOI City Mall	2018	1
SEDAN	KLIA	2018	1
SEDAN	KLIA	2019	1
SEDAN	KPJ Hospital	2019	1
SEDAN	MMU	2019	1
SEDAN	Tune hotel	2018	1
SEDAN		2018	3
SEDAN		2019	4
SUV	CIMB Bank	2018	1
SUV	Gurney Plaza	2019	1
SUV	The Mines	2018	1
SUV		2018	2
SUV		2019	1

c. 1 SQL each on the following operator:

- Comparison operators: =, <, <=, >=, <>

We used the above operators to show customer ID that is not equal to “C0001” and “C0014”, Driver_ID that is not equal to “DR01”, trip_price more than 10 and less than 60 and points_earned more than or equal to 20

```
SELECT t.cus_id,
       cus_name,
       driver_id,
       trip_price,
       points_earned
FROM   trips t,
       customer c
WHERE  t.cus_id = c.cus_id
       AND t.cus_id <> 'C0001'
       AND t.cus_id <> 'C0014'
       AND t.driver_id <> 'DR01'
       AND t.trip_price > 10
       AND trip_price < 60
       AND t.points_earned >= 20;
```

CUS_ID	CUS_NAME	DRIVER_ID	TRIP_PRICE	POINTS_EARNED
C0002	Isaac Smith	DR08	17.70	26
C0002	Isaac Smith	DR05	28.40	56
C0003	John Cena	DR07	17.78	26
C0007	Tanveer Ahmad	DR11	29.30	58
C0008	Ling Hui	DR17	17.40	26
C0009	Mahedi Munem	DR19	24.00	48
C0011	Muhaimenul Abir	DR09	47.78	143
C0015	Joy Das	DR16	15.40	23
C0020	Ellen Ripley	DR02	15.40	23

- Logical operator OR, AND, NOT

List of customers and number of trips they took to Cyberjaya or Putrajaya

```
SELECT cus_name,
       loc_city,
       Count(trip_id) AS NumOfTrip
FROM   customer,
       trips,
       location
WHERE  customer.cus_id = trips.cus_id
      AND trips.loc_id = location.loc_id
      AND ( loc_city = 'Cyberjaya'
           OR loc_city = 'Putrajaya' )
GROUP BY cus_name,
         loc_city
ORDER BY loc_city ASC;
```

CUS_NAME	LOC_CITY	NUMOFTRIP
Muhaimenul Abir	Cyberjaya	1
Tanveer Ahmad	Cyberjaya	2
Thomas Anderson	Cyberjaya	1
Ellen Ripley	Putrajaya	1
Isaac Smith	Putrajaya	1
Marie Curie	Putrajaya	1
Ridwan Mahir	Putrajaya	1

- Special operator BETWEEN, IS NULL, LIKE, IN, EXISTS

This selects the driver name that has the letter 'M/m' in their name and the city he/she visited between January and July

```
SELECT driver_name,
       Monthname(booking_time) AS MONTH,
       loc_city
FROM   driver d,
       time t,
       location l,
       trips p
WHERE  d.driver_id = p.driver_id
      AND t.time_id = p.time_id
      AND l.loc_id = p.loc_id
      AND t.time_month BETWEEN 1 AND 7
      AND ( d.driver_name LIKE '%m%'
            OR d.driver_name LIKE '%M%' );
```

DRIVER_NAME	MONTH	LOC_CITY
Roosevelt Merry	January	Bukit Jalil
Roosevelt Merry	July	Selangor
Mandy Stewart	April	Penang
Munira Hossain	May	Putrajaya
Jasmine Sikder	May	Putrajaya
Roosevelt Merry	March	Putrajaya
Tamzid Hossain	May	Putrajaya
Michael Stevens	June	Ampang

- Aggregate operator COUNT, MAX, MIN, SUM, AVG

List sum of all the driver trip, to know the top earn driver for the first half of the year

```
SELECT r.driver_id, d.driver_name,
       Sum(trip_price) AS total,
       Count(r.driver_id) AS no_of_trip
FROM   driver d,
       trips r,
       time e
WHERE  r.time_id = e.time_id
       AND d.driver_id = r.driver_id
       AND e.time_month BETWEEN 1 AND 7
GROUP BY r.driver_id, driver_name
ORDER BY total DESC;
```

DRIVER_ID	DRIVER_NAME	TOTAL	NO_OF_TRIP
DR09	John Kok	47.78	1
DR14	Jasmine Sikder	36.90	1
DR19	Farooq Sheikh	36.00	2
DR01	Roosevelt Merry	34.10	3
DR11	Yan Gun	29.30	1
DR05	Munira Hossain	28.40	1
DR15	Jahid Hasan	18.00	1
DR07	Mandy Stewart	17.78	1
DR02	Tamzid Hossain	15.40	1
DR16	Jewel Hasan	15.40	1
DR03	Michael Stevens	7.99	1

d. View

This view selects customers who never took any trip.

```
CREATE VIEW cusnotrip AS
SELECT cus_id, cus_name
FROM customer
WHERE cus_id NOT IN (SELECT cus_id FROM trips);

SELECT * FROM cusnotrip;
```

CUS_ID	CUS_NAME
C0012	Bulbul Alam
C0013	Edward George
C0017	Johnny Depp
C0021	Heath Ledger
C0022	Forrest Gump

e. TWO SQL not covered in lecture

1) Any

The Any operator is used with a WHERE or HAVING clause. The ANY operator returns true if any of the subquery values meet the condition. The following query list customer names where customers took a trip charged more than 40RM.

```
SELECT cus_name
FROM customer
WHERE cus_id = ANY (SELECT cus_id FROM trips WHERE trip_price > 40);
```

CUS_NAME
Thomas Anderson
Sasmitta Krishnan
Muhaimenul Abir

2) Decode

Decode is a function in Oracle and is used to provide if-then-else type of logic to SQL, in this statement, it checks in the location table in the first 3 columns for the city names and when found it gives a result which is here shown as the state which the city is in.

```
SELECT loc_city, Decode (loc_city,
'Cyberjaya','SGR',
'Bukit Jalil','KL',
'Kuala Lumpur','KL') State
FROM location
LIMIT 3;
```

LOC_CITY	STATE
Cyberjaya	SGR
Bukit Jalil	KL
Kuala Lumpur	KL