ISM3212C DATABASE MANAGEMENT ONLINE

Module 7 Activities

Scenario and Database Model: InstantRide

InstantRide is the new ride sharing application in the city and it has just started its operations. With the help of the InstantRide mobile application, the users request a ride with their location. Drivers and cars are assigned to the request; and then the driver picks up the user to ride their requested location. Information for the users, drivers and cars are stored in the database as well as the travel transactions.

In the USERS table, information for the users are stored with their first name, last name and email:

USER_ID	USER_FIRST_NAME	USER_LAST_NAME	USER_EMAIL
3001	Jack	Hill	j.hill@xmail.com
3002	Ryan	Collins	r.collins@xmail.com
3003	Nursin	Yilmaz	n.atak@gmail.com
3004	Sarah	Price	s.price@xmail.com
3005	Bobby	Griffin	b.griffin@xmail.com
3006	Randy	Clark	r.clark@xmail.com
3007	Jose	Thomas	j.thomas@xmail.com
3008	Nursin	Yilmaz	n.yilmaz@xmail.com

USERS Table

In the DRIVERS table, all the drivers in the InstantRide are stored with their name, driving license number and check and rating information:

DRIVER_ID	DRIVER_FIRST_NAME	DRIVER_LAST_NAME	DRIVER_DRIVING_LICENSE_ID	DRIVER_START_DATE	DRIVER_DRIVING_LICENSE_CHECKED	DRIVER_RATING
2001	Willie	Butler	1874501	2019-09-12	1	4.4
2002	Justin	Howard	1953853	2019-09-09	1	4.8
2003	Anthony	Walker	1735487	2019-09-15	1	3.5
2004	Ece	Yilmaz	1734747	2019-08-15	1	0

DRIVERS Table

In the CARS table, all the cars in the InstantRide system are kept with the license plate, model and year:

CAR_ID	CAR_PLATE	CAR_MODEL	CAR_YEAR
1001	BB-542-AB	TOYOTA PRIUS	2018
1002	BB-883-EE	TESLA MODEL 3	2019
1003	BB-451-ZN	TOYOTA AURIS	2019
1004	BB-189-MM	MERCEDES E200	2019

CARS Table

Finally, the transactions of the rides are stored in the TRAVELS table. For each travel, start and end time with location are stored. In addition, the involved driver, car and user are listed for each drive. Price and discount information are also available in the database:

TRAVEL_ID	TRAVEL_START_TIME	TRAVEL_END_TIME	TRAVEL_START_LOCATION	TRAVEL_END_LOCATION	TRAVEL_PRICE	DRIVER_ID	CAR_ID	USER_ID	TRAVEL_DISCOUNT
5001	2019-10-01 04:04:55	2019-10-01 04:14:19	9614 York Road	84 Church Lane	15.44	2001	1003	3005	NULL
5002	2019-10-01 05:57:33	2019-10-01 06:12:33	47 Church Street	68 High Street	20.56	2001	1003	3006	NULL
5003	2019-10-01 13:35:20	2019-10-01 13:45:10	2 Windsor Road	95 West Street	12.32	2002	1001	3002	NULL
5004	2019-10-02 08:44:48	2019-10-02 09:15:28	9060 Mill Lane	27 Main Road	30.49	2003	1002	3001	0.13
5005	2019-10-02 16:38:54	2019-10-02 16:48:10	2 Queensway	24 Mill Lane	11.15	2001	1003	3007	NULL
5006	2019-10-03 19:12:14	2019-10-03 19:23:45	50 Main Road	93 Broadway	14.61	2003	1002	3007	0.10
5007	2019-10-03 16:06:36	2019-10-03 16:08:56	39 Park Road	91 West Street	4.41	2002	1004	3003	0.14
5008	2019-10-03 17:17:12	2019-10-03 17:37:42	37 The Drive	17 Stanley Road	25.12	2001	1003	3001	0.25
5009	2019-10-03 21:16:48	2019-10-03 21:26:18	77 Mill Road	724 Springfield Road	13.55	2001	1003	3005	NULL
5010	2019-10-03 23:21:40	2019-10-03 23:39:10	16 Church Road	30 North Road	25.62	2003	1002	3003	0.20

TRAVELS Table

You are assigned as the database administrator to collect and manage transactional data of the InstantRide operations. Your main task is to create SQL scripts to help other teams to retrieve the requested data. In the following activities, you will create the scripts, run against the database and send the result to the corresponding teams.

Grading

After you have completed a problem and clicked the **Run Query** button, mark the task as complete. Checks will run to verify your work.

When all problems are completed and you are satisfied with the results, use the **Submit** button to record your score.

ACTIVITY 1

Task 1:

Drivers are essential for InstantRide, and the Driver Relationship team is responsible for their integration and success. The team requires all the driver detail in the system for creating a new dashboard. You need to **SELECT** all available data for the drivers and return back to the team.

Task 2:

The Driver Relationship team also requests the joining dates of the drivers to create a timeline. In the table, you only need to return the joining date of the drivers. You need to only return the **DRIVER_START_DATE** column inside a **SELECT** statement for the `DRIVERS` table.

Task 3:

The Driver Relationship team requires the following details about the drivers:

- All drivers with their rating in descending order
- All drivers currently having a rating higher than **4**

You need to return the **DRIVER_ID** and **DRIVER_RATING** couples in two separate tables.

Task 4:

The InstantRide User Satisfaction team is a core team for InstantRide, and they focus on increasing the customer satisfaction. They want to learn the travel time for each ride in the system. You need to return the **USER_ID**, and the **TRAVEL_TIME** column which is calculated using the `TIMEDIFF` function on the `TRAVEL_END_TIME` and the `TRAVEL_START_TIME`.

Task 5:

User Satisfaction team wants to send monthly summaries for each user. They need the following details with the user ID:

- The last day of the month when the users traveled most recently
- One week after the last day of the month when the users traveled most recently

You need to return a three-column output
with **USER_ID**, **LAST_TRAVEL_MONTH** and **NOTIFICATION**. `LAST_TRAVEL_MONTH` should be calculated using the **MAX** of the `LAST_DAY` of the `TRAVEL_END_TIME` field.
Similarly, `NOTIFICATION` should be calculated with `DATE_ADD` function to add one week.

Task 6:

The Marketing team of InstantRide wants to know that how many discounts have been offered for each ride. You need to calculate this information for each travel where a discount is applied and return two columns: **TRAVEL_ID** and **DISCOUNT_AMOUNT**. In addition, you need to return the calculation as a money value using the `ROUND` function to **2** decimals.

QUERIES

```
/*TASK 1*/
SELECT
   *
FROM
   DRIVERS;

/*TASK 2*/
SELECT
   DRIVER_START_DATE
FROM
   DRIVERS;

/*TASK 3*/
SELECT
   DRIVER_ID, DRIVER_RATING
FROM
   DRIVERS
ORDER BY DRIVER_RATING DESC;

/*TASK 3 - 2nd TABLE*/
SELECT
   DRIVER_ID, DRIVER_RATING
FROM
   DRIVERS
WHERE
   DRIVER_RATING > 4
ORDER BY DRIVER_RATING DESC;
```

```
SELECT
   USER_ID, TIMEDIFF(TRAVEL_END_TIME, TRAVEL_START_TIME)
    AS TRAVEL_TIME
FROM
   TRAVELS;

/*TASK 5*/
SELECT
   USER_ID,
   MAX(LAST_DAY(TRAVEL_END_TIME)) AS LAST_TRAVEL_MONTH,
   DATE_ADD(MAX(LAST_DAY(TRAVEL_END_TIME)), INTERVAL 1 WEEK) AS NOTIFICATION
FROM
   TRAVELS
GROUP BY USER_ID;

/*TASK 6*/
SELECT TRAVEL_ID,
      ROUND(TRAVEL_PRICE * TRAVEL_DISCOUNT, 2) AS DISCOUNT_AMOUNT
FROM
   TRAVELS
WHERE
   TRAVEL_DISCOUNT IS NOT NULL;
```

ACTIVITY 2

Task 1:

The InstantRide received some traffic violation tickets from the government. The Legal team of InstantRide requires the travel information of the respective drivers along with corresponding Driving License IDs to proceed further. In addition, the team wants to include the drivers without travel information in the system yet for the completion of driver list. Therefore, you need to
return **DRIVER_FIRST_NAME**, **DRIVER_LAST_NAME**, **DRIVER_DRIVING_LICENSE_ID**, **TRAVEL_START_TIME**, **TRAVEL_END_TIME** information from the `DRIVERS` and `TRAVELS` data connected by `LEFT JOIN`.

Task 2:

The InstantRide Management team considers setting up a Lost & Found inventory. In order to start the setup, the team requires the detail of users with their travel start and end times. The team wants to track potential list of users who may have forgotten their items on the cars. Therefore, you need to
return **USER_FIRST_NAME**, **USER_LAST_NAME**, **TRAVEL_START_TIME**, **TRAVEL_END_TIME** information from the `USERS` and `TRAVELS` tables connected inside a `JOIN` statement by the `USING` function and `USER_ID` field.

Task 3:
The InstantRide Finance team wants to collect the price and discount information with the driver names for each travel in the system. You need to return the **TRAVEL_ID**, **DRIVER_FIRST_NAME**, **DRIVER_LAST_NAME**, **TRAVEL_PRICE**, and **TRAVEL_DISCOUNT** information from the `TRAVELS` and `DRIVERS` tables combined over `DRIVER_ID` field with the `ON` keyword.

Task 4:
The InstantRide Driver Relationship team wants to create groups for drivers according to their ratings such as 3+ or 4+. For instance, a driver with the rating *3.8* will be *3+*; whereas a driver with the rating *4.2* will be *4+*. You need to return a two column output with **DRIVER_ID** and **DRIVER_RATING** which has first `FLOOR` applied and then `CONCAT` with `+` sign for all drivers with a rating greater than **0**.

Task 5:
The InstantRide User Satisfaction team are looking forward to creating discounts for the users. However, the team suspects that there could be duplicate users in the system with different emails. Check for the users with their names and surnames for potential duplicates. Therefore, you need to `JOIN` the `USERS` table with `USERS` table and compare for equality of `USER_FIRST_NAME` and `USER_LAST_NAME` and difference in `USER_ID` fields.

QUERIES

```
/*TASK 1*/
SELECT
   DRIVER_FIRST_NAME, DRIVER_LAST_NAME, DRIVER_DRIVING_LICENSE_ID,
   TRAVEL_START_TIME, TRAVEL_END_TIME
FROM
   DRIVERS D
      LEFT JOIN
   TRAVELS T ON D.DRIVER_ID = T.DRIVER_ID;

/*TASK 2*/
SELECT
   USER_FIRST_NAME, USER_LAST_NAME,
   TRAVEL_START_TIME, TRAVEL_END_TIME
FROM
   USERS
      JOIN
   TRAVELS USING (USER_ID);

/*TASK 3*/
SELECT
   TRAVEL_ID, D.DRIVER_FIRST_NAME, D.DRIVER_LAST_NAME,
```

```
    TRAVEL_PRICE, TRAVEL_DISCOUNT
FROM
    TRAVELS T
        JOIN
    DRIVERS D ON T.DRIVER_ID = D.DRIVER_ID;

/*TASK 4*/
SELECT
    DRIVER_ID, CONCAT(FLOOR(DRIVER_RATING), "+") AS DRIVER_RATING
FROM
    DRIVERS
WHERE DRIVER_RATING >= 1.0;

/*TASK 5*/
SELECT
    *
FROM
    USERS U
        JOIN
    USERS B ON U.USER_FIRST_NAME = B.USER_FIRST_NAME
        AND U.USER_LAST_NAME = B.USER_LAST_NAME
        AND U.USER_ID != B.USER_ID;
```

ACTIVITY 3

Task 1:

The InstantRide Driver Relationship team wants to analyze the travel information of the low rated drivers. You will need to provide them with all the travel information of the drivers with the average rating lower than **4**. The team wants to get in touch with the travelers and analyze their feedback. You need to run **SELECT** query and return all travel data from `TRAVELS` table filtered by the drivers who has lower rating than **4** in the `DRIVERS` table.

Task 2:

The InstantRide Driver Relationship team wants to check if there are any drivers with zero rides. You need to extract the **DRIVER_ID**, **DRIVER_FIRST_NAME**, **DRIVER_LAST_NAME** of the drivers with zero rides. You can use a subquery with `DRIVER_ID` compared to `ALL` rows in `TRAVELS`.

Task 3:

The InstantRide Finance team wants to know the average discount amounts for each car in the InstantRide. Calculate the average discount amount as monetary value for the travels where a discount is applied. You need to create a subquery over the `TRAVELS` table to

retrieve **CAR_ID** and **DISCOUNT_AMOUNT**, calculated with **2** decimals using the `ROUND` function.
To calculate the `DISCOUNT_AMOUNT`, multiply the `TRAVEL_PRICE` by the `TRAVEL_DISCOUNT` where the `TRAVEL_DISCOUNT` value is not `NULL`. Round the result to **2** decimals.
Then you can use this subquery to get the **CAR_ID** and **AVG** of `DISCOUNT_AMOUNT` values, once again using the `ROUND` function on the average results. Group the results by the `CAR_ID`. Use **CAR_ID** and **DISCOUNT_AMOUNT** as column aliases and return it back to the Finance team.

Task 4:

The InstantRide Finance team also wants to analyze travels where more than the average discount rate is applied. They want to look for any correlation between higher discount amounts against other travel characteristics. You need to create a **SELECT** statement which is filtered with a subquery to calculate the **AVG** of the `TRAVEL_DISCOUNT` column.

Task 5:

The InstantRide Management team considers creating a new team for Car Maintenance. The new team needs to find/list the cars that are used more than average with the usage count. Collect the information of all rides and consolidate over the Car IDs. You need to create a three level SQL statement. Firstly, you need to **COUNT** the number of rows in `TRAVELS` and **GROUP_BY** the `CAR_ID` field. Then you need to calculate the **AVG** of the data to find the average usage of the cars. Finally, you need to return **CAR_ID** and the `TRAVELS` count (as the **Usages** column) filtered to only values greater than the calculated average.

Task 6:

The InstantRide Marketing team wants to organize an InstantRide party. The team requires **first name** and **last name** of all the users and drivers in order to create a gate-pass for their entry. You need to join `USERS` and `DRIVERS` tables using `UNION` and return `FIRST_NAME` and `LAST_NAME` columns.

QUERIES

```
/*TASK 1*/
SELECT
   *
FROM
   TRAVELS
WHERE
   DRIVER_ID IN (SELECT
         DRIVER_ID
      FROM
         DRIVERS
```

```
        WHERE
            DRIVER_RATING < 4);

/*TASK 2*/
SELECT
    D.DRIVER_ID, D.DRIVER_FIRST_NAME, D.DRIVER_LAST_NAME
FROM
    DRIVERS D
WHERE
    D.DRIVER_ID != ALL (SELECT DISTINCT
            T.DRIVER_ID
        FROM
            TRAVELS T)

/*TASK 3*/
SELECT
    CAR_ID,
    ROUND(AVG(DISCOUNT), 2) AS DISCOUNT_AMOUNT
FROM
    (SELECT
        CAR_ID,
        ROUND(TRAVEL_PRICE * TRAVEL_DISCOUNT, 2) AS DISCOUNT
    FROM
        TRAVELS
    WHERE
        TRAVEL_DISCOUNT IS NOT NULL) AS TEMP_TABLE
GROUP BY
    CAR_ID;

/*TASK 4*/
SELECT
    *
FROM
    TRAVELS
WHERE
    TRAVEL_DISCOUNT > (SELECT
            AVG(TRAVEL_DISCOUNT)
        FROM
            TRAVELS);

/*TASK 5*/
SELECT
    CAR_ID, COUNT(TRAVEL_ID) AS Usages
```

```
FROM
   TRAVELS
GROUP BY CAR_ID
HAVING COUNT(TRAVEL_ID) > (SELECT
         AVG(U.USAGES)
      FROM
         (SELECT
            COUNT(TRAVEL_ID) AS Usages
         FROM
            TRAVELS
         GROUP BY CAR_ID) AS U);

/*TASK 6*/
SELECT
   USER_FIRST_NAME as FIRST_NAME,
   USER_LAST_NAME as LAST_NAME
FROM
   USERS
UNION
SELECT
   DRIVER_FIRST_NAME as FIRST_NAME,
   DRIVER_LAST_NAME as LAST_NAME
FROM
   DRIVERS;
```

ACTIVITY 4

Task 1:

The InstantRide Driver Relationship team wants the details of the drivers and the number of cars they use. Calculate the count of the **CAR_ID** field in the `TRAVELS` table and return a table grouped by **DRIVER_ID** and in descending order by the calculated `COUNT` column, displayed as **CARS**.

Task 2:

The InstantRide Driver Relationship team requires the detail of the drivers who has used more than one car for rides more than once. Send them the **DRIVER_ID** with the calculated `COUNT` of used cars, displayed as **CARS**, in decreasing order.

Task 3:

The InstantRide Marketing team is planning to create a heatmap of the start locations of the travels in InstantRide. The team requires the **DISTINCT TRAVEL_START_LOCATION** values for all travels in the system.

Task 4:
The InstantRide User Satisfaction team requires the average and maximum number of rides users have taken so far with InstantRide. In addition, they would like to know the total number of travels. However, they need these details with the corresponding column names **Average**, **Maximum** and **Total** by using the `AVG`, `MAX` and `SUM` functions. In order to accomplish this, you will first need to create a derived table from the `TRAVELS` table to pass the `TRAVEL_ID` count to the three mathematical functions.

Task 5:
The InstantRide Driver Relationship team wants to learn how many travels each driver has done in the month of October. You need to send them the **DRIVER_ID**, and two calculated columns: **DAY** and **RIDES**. The `DAY` column is calculated using the `DAY()` function with the `TRAVEL_START_TIME` as the argument. The `RIDES` column is calculated by using the `COUNT()` function to determine the number of rides given for each day. Filter the results with the `MONTH` function.

Task 6:
The InstantRide User Satisfaction team received a phone call from a user who might have left her wallet in the car. She indicated that the license plate of the car was starting with *BB-883* but unfortunately, she could not remember the full license plate number. The team wants to get all travel information for the cars with the license plate starting with **BB-883**. You need to return all travel data from the `TRAVELS` table for the `CAR_ID` which has a plate number compared with `SUBSTR` and `UPPER` functions.

QUERIES

```
/*TASK 1*/
SELECT
   DRIVER_ID, count(CAR_ID) AS CARS
FROM
   TRAVELS
GROUP BY (DRIVER_ID)
ORDER BY CARS DESC;

/*TASK 2*/
SELECT
   DRIVER_ID, COUNT(CAR_ID) AS CARS
FROM
   TRAVELS
GROUP BY DRIVER_ID
HAVING COUNT(CAR_ID) > 1
ORDER BY COUNT(CAR_ID) DESC;

/*TASK 3*/
```

```sql
SELECT DISTINCT
    TRAVEL_START_LOCATION
FROM
    TRAVELS;

/*TASK 4*/
SELECT
    AVG(TravelTotal) AS Average,
    MAX(TravelTotal) AS Maximum,
    SUM(TravelTotal) AS Total
FROM
(SELECT USER_ID, COUNT(TRAVEL_ID) AS TravelTotal /*Based on user_id get the TravelCount*/
    FROM
        TRAVELS
    GROUP BY USER_ID) AS TEMP_TABLE;

/*TASK 5*/
SELECT
    DRIVER_ID, DAY(TRAVEL_START_TIME) AS DAY, COUNT(TRAVEL_ID) AS RIDES
FROM
    TRAVELS
WHERE
    MONTH(TRAVEL_START_TIME) = 10
GROUP BY DRIVER_ID, DAY(TRAVEL_START_TIME);

/*TASK 6*/
SELECT
    *
FROM
    TRAVELS
WHERE
  CAR_ID IN (SELECT
              CAR_ID
          FROM
              CARS
          WHERE
              UPPER(SUBSTR(CAR_PLATE,1,6)) = 'BB-883');
```