INFS1200/7900 Assignment 2

Code Due: 4 October 2024 @ 3:00 PM AEST
Oral Assessment: Week 12, 14-18 October 2024

Weighting: 25%

Full Name	Student ID (8 digits)
Abdallah Azazy	47994832

Overview

The purpose of this assignment is to test your ability to use and apply SQL concepts to complete tasks in a real-world scenario. Specifically, this assessment will examine your ability to use SQL Data Manipulation Language to return specific subsets of information that exist in a database and Data Definition Language to create a new relational schema. The assignment is to be completed **individually.**

Submission

Assignment 2 is made up of two parts. **Part 1** will be submitted through an electronic marking tool called Gradescope, which will also be used for providing feedback. **Part 2** is an oral assessment that will be completed during an in-person meeting with a tutor during a practical session in Week 12 (after your Gradescope submission). Details below:

Part 1: Answer the questions on this task sheet and submit them through an electronic marking tool called Gradescope. For this assignment, you will need to submit two types of files to the portal:

- **Query Files**:
  - For each question in Sections A, B and C, you are required to submit a separate *.sql* or *.txt* file which contains your SQL query solution for that question (submit only one of these files; if you submit both, the *.sql* file will be graded).
  - Each file should only contain the SQL query(s) and no additional text.
  - Each file should be named as per the *Filename* description in the question.
  - The total number of queries allowed to be run per question is also specified in each question's description.
  - When submitting files to the autograder, select all of your *.sql* or *.txt* files as well as your *.pdf* file.

- **Assignment PDF**:
  - Insert your answers for all Sections A-D into the template boxes on this assignment task sheet where appropriate, then export this document to a PDF and also upload it to the Gradescope autograder portal.
  - Only Section D will be hand-marked from your PDF submission, however this is also a backup for Sections A, B and C in case of autograder failure.
  - For Sections A, B and C, include a screenshot of the output of your query for each question in the space provided. Use your zones to generate the output.
  - For queries with a returning relation of more than 10 tuples, you can use a **`LIMIT 10`** clause to only capture the first 10 tuples of the table. Only use **`LIMIT 10`** to get a screenshot of your output for the pdf submission, don't include it in your code submission.
  - **Please name your file 'Assignment_2.pdf'.** Please do not alter the format or layout of this document and ensure the name and SID boxes are completed.

Part 2 is an oral assessment, to verify your understanding of the code you submitted in Part 1 Sections A, B and C.

- This will be an oral critique of your submitted code. In a short meeting with a member of the teaching staff during Week 12 practical sessions, you will explain the work you have submitted in Part 1 and discuss your choices.
- All oral assessments must be given live and will be recorded by the teaching team (i.e. on Zoom) for archiving purposes.

Marking

Assignment 2 is worth **25 course marks**, and marking is made up of two parts.

First, the marks available per section of Part 1 are as follows (note that INFS1200 differs from INFS7900):

	INFS1200	**INFS7900**
Section A – SQL DML (SELECT)	15 marks	13 marks
Section B – SQL DML (UPDATE, INSERT, DELETE)	4 marks	4 marks
Section C – SQL DDL	4 marks	4 marks
Section D – Critical thinking	2 marks	4 marks

Given these available marks, **students must also achieve a pass (+/-) in Part 2**, the oral critique, to be eligible to pass Assignment 2. Failure in Part 2 will result in your mark being capped at 12.5%.

Grading and autograder feedback

Sections A, B and C of this assignment will be graded via an autograder deployed on Gradescope. However, we reserve the right to revert to hand marking using the pdf submission should the need arise.

Specifically, your assignment may be graded against several data instances, which may include a simple (and small) data instance, a large data instance or instances containing curated edge cases. The correctness of your queries will be judged by comparing your queries' return values to those of our solutions, because there is usually more than one equivalent way to execute a given query.

Note that solutions to each question will be limited to contain a maximum of 4 queries.

When you submit your code, the autograder will provide you with two forms of immediate feedback:

- **File existence and compilation tests**: Your code will be checked to see if it compiles correctly. If it fails one or more compilation test, the errors returned by the autograder will help you debug. Note that code that fails to compile will receive 0 marks. No marks are given for passing the compilation tests.
- **Simple instance data tests for Section A:** The autograder will return your degree of success on the simple data instance for the queries in Section A, so that you can judge your progress (i.e. 9/10 simple instance tests passed). Individual test results will not be revealed, and your submission's performance on the more difficult instances will remain hidden until grades are released. Final weightings on the different test instances will also remain hidden until grades are released.

More details will be provided regarding how you can interpret the results of these tests and what it means for your assignment grade during practicals.

Note: Your queries must compile using **MySQL version 8.0**. This is the same DBMS software as is used on your zones. You may use any MySQL function that have been used in class in addition to those specified in the questions. You may also use other MySQL functions not covered in this course to assist with manipulating the data if needed, however please ensure you read the MySQL documentation page first to ensure the functions works as intended.

The final details of the Gradescope autograder will be released closer to the assignment deadline. Note that you will be able to resubmit to the autograder an unlimited number of times before the deadline.

Materials provided:

You will be provided with the database schema and a simple data instance that .

Because the autograder uses the same DBMS as your zones, you are encouraged to use your zones to develop your assignment answers.

Late penalties: Please consult the course profile for late penalties that apply to this assessment item.

Plagiarism

The University has strict policies regarding plagiarism. Penalties for engaging in unacceptable behaviour range from loss of grades in a course through to expulsion from UQ. You are required to read and understand the policies on academic integrity and plagiarism in the course profile (Section 6.1). If you have any questions regarding an acceptable level of collaboration with your peers, please see either the lecturer or your tutor for guidance. Remember that ignorance is not a defence!

You are permitted to use generative AI tools to help you complete this assessment task. However, if you do, please provide complete copies of your interactions with the AI tool in the space provided at the end of your submission. Please note that if you use generative AI but fail to acknowledge this by attaching your interaction to the end of the assignment, it will be considered misconduct, as you are claiming credit for work that is not your own.

Task

For this assignment, you will be presented with the simplified schema of a car insurance company.

EasyDrive Insurance is a direct-to-consumer insurance company dedicated to providing affordable and competitive car insurance to their customers by gathering extensive and meaningful information about their customers and the vehicles they drive.

When a customer navigates to the EasyDrive Insurance website, they are first required to create a profile, capturing personal details stored in the *Customers* and *Address* table. A customer may then choose to purchase an insurance policy for their vehicle by filling out a questionnaire detailing their personal information and the usage of their car. Customers also have the option to insure their vehicle over consecutive years. For several years, EasyDrive Insurance has operated through a low-fidelity website, efficiently selling car insurance policies and storing customer data in a relational database management system (DBMS) with the following schema:

- **Customers** table records customer-specific information.
- **Address** table stores the addresses of the customers and is linked to the Customers table.
- **Vehicle** table records information about the vehicles insured by the Customer.
- **VehicleCodeMapping** is a central table used to define a VehicleCode, which is based upon a vehicles' Make, Model and Year. VehicleCodes are further used to discern the acceptable excess range and value of a vehicle.
- **VehicleExcessRange** defines the minimum and maximum excess that would be allowed for each VehicleCode.
- **VehicleValue** outlines the redeemable market value (also known as the sum insured) of the vehicle via its VehicleCode. The customer is paid out the redeemable market value if the vehicle were to be in an accident.
- **Policy** records the insurance policy purchased by a customer for their vehicle, in a given policy year.

Relational Schema:

Customer [<u>CustomerID</u>, Name, DateOfBirth, Email, Occupation, AddressID]

Address [<u>AddressID</u>, StreetName, Number, Suburb, Postcode, State, Country]

Vehicle [<u>VehicleID</u>, VehicleCode, VehiclePurpose, EstYearlyKm]

VehicleCodeMapping [<u>VehicleCode</u>, Make, Model, Year]

VehicleValue [<u>VehicleCode</u>, MarketValue]

VehicleExcessRange [<u>VehicleCode</u>, MinimumExcess, MaximumExcess]

Policy [<u>PolicyID,</u> CustomerID, VehicleID, PolicyStartYear, PolicyPurchaseDate, Excess, Premium]

Foreign Keys:

Customer.AddressID references Address.AddressID

Policy.VehicleID references Vehicle.VehicleID

Policy.CustomerID references Customer.CustomerID

Vehicle.VehicleCode references VehicleCodeMapping.VehicleCode

VehicleExcessRange.VehicleCode references VehicleCodeMapping.VehicleCode

VehicleValue.VehicleCode references VehicleCodeMapping.VehicleCode

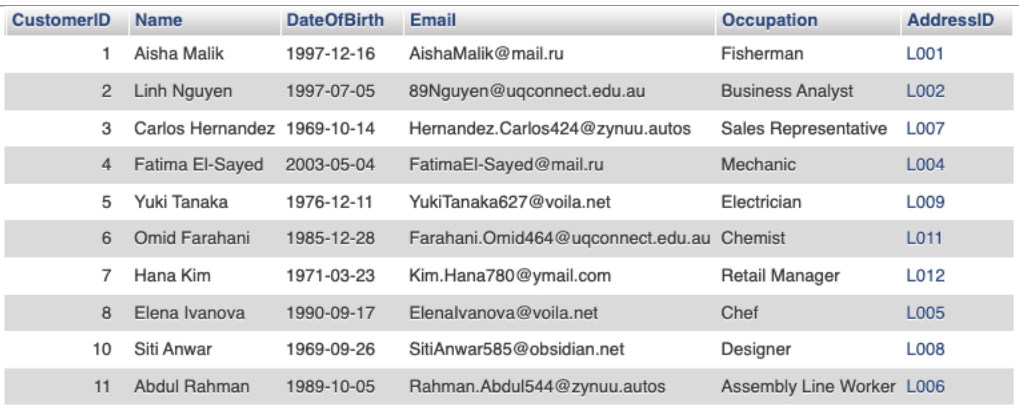
The Entity-Relationship Diagram (ERD) is provided for this schema in Appendix 1.

For this assignment you will be required to write SQL queries to answer to complete the tasks below.

- Answer the queries using only the information provided in the **Task** box.
- Use the **SQL Solution** box provided to record your answer code.
- Use the **Output Screenshot** box to record the output of your query (generated in your zones before submission). For queries with a returning relation of more than 10 tuples, you can use the **LIMIT 10** clause to only capture the first 10 tuples of the table for your output screenshot.

Example Query	
Task	Return all Customers

SQL Solution	SELECT *
	FROM Customer;

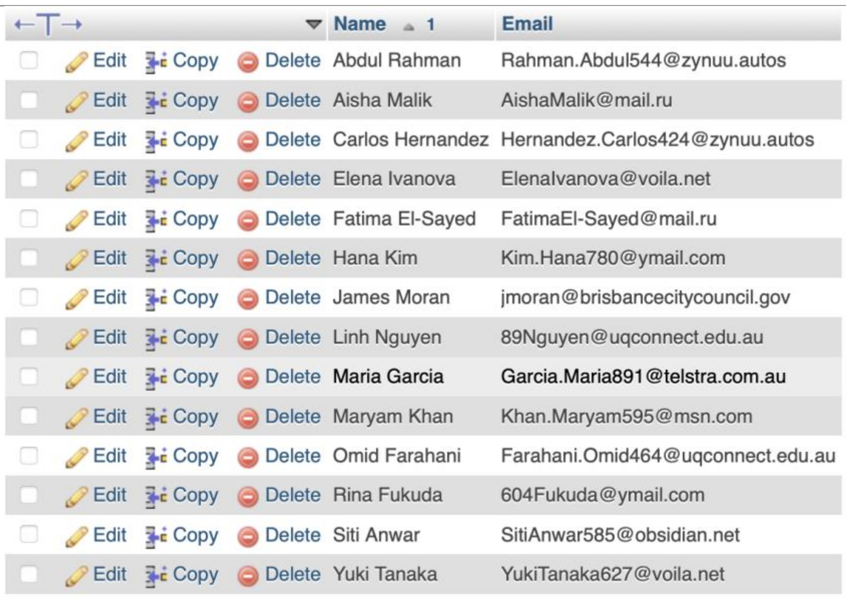
Output Screenshot	

CustomerID	Name	DateOfBirth	Email	Occupation	AddressID
1	Aisha Malik	1997-12-16	AishaMalik@mail.ru	Fisherman	L001
2	Linh Nguyen	1997-07-05	89Nguyen@uqconnect.edu.au	Business Analyst	L002
3	Carlos Hernandez	1969-10-14	Hernandez.Carlos424@zynuu.autos	Sales Representative	L007
4	Fatima El-Sayed	2003-05-04	FatimaEl-Sayed@mail.ru	Mechanic	L004
5	Yuki Tanaka	1976-12-11	YukiTanaka627@voila.net	Electrician	L009
6	Omid Farahani	1985-12-28	Farahani.Omid464@uqconnect.edu.au	Chemist	L011
7	Hana Kim	1971-03-23	Kim.Hana780@ymail.com	Retail Manager	L012
8	Elena Ivanova	1990-09-17	ElenaIvanova@voila.net	Chef	L005
10	Siti Anwar	1969-09-26	SitiAnwar585@obsidian.net	Designer	L008
11	Abdul Rahman	1989-10-05	Rahman.Abdul544@zynuu.autos	Assembly Line Worker	L006

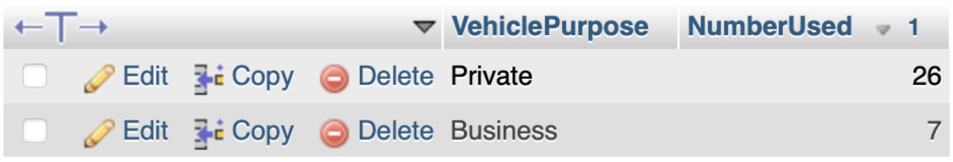
Section A – SQL DML (SELECT)

Question 1	
Task	Return the distinct name and email of all customers, ordered in alphabetical order of their name.

SQL Solution	`SELECT DISTINCT Name, Email`
	`FROM Customer`
	`ORDER BY Name;`

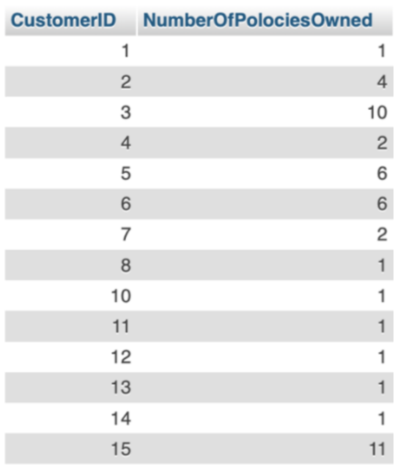
Output Screenshot	

Name	Email
Abdul Rahman	Rahman.Abdul544@zynuu.autos
Aisha Malik	AishaMalik@mail.ru
Carlos Hernandez	Hernandez.Carlos424@zynuu.autos
Elena Ivanova	ElenaIvanova@voila.net
Fatima El-Sayed	FatimaEl-Sayed@mail.ru
Hana Kim	Kim.Hana780@ymail.com
James Moran	jmoran@brisbancecitycouncil.gov
Linh Nguyen	89Nguyen@uqconnect.edu.au
Maria Garcia	Garcia.Maria891@telstra.com.au
Maryam Khan	Khan.Maryam595@msn.com
Omid Farahani	Farahani.Omid464@uqconnect.edu.au
Rina Fukuda	604Fukuda@ymail.com
Siti Anwar	SitiAnwar585@obsidian.net
Yuki Tanaka	YukiTanaka627@voila.net

Question 2	
Task	Return the number of vehicles used for every type of Vehicle Purpose available in the database, ordered from greatest to least.
Explanation	This query should return two columns, one for the VehiclePurpose, and one for the number of vehicles insured under each VehiclePurpose.
SQL Solution	`SELECT VehiclePurpose, Count(*) as NumberUsed` `FROM Vehicle` `GROUP BY VehiclePurpose` `ORDER BY NumberUsed DESC;`
Output Screenshot	

```sql
SELECT VehiclePurpose, Count(*) as NumberUsed
FROM Vehicle
GROUP BY VehiclePurpose
ORDER BY NumberUsed DESC;
```

	VehiclePurpose	NumberUsed
Edit Copy Delete	Private	26
Edit Copy Delete	Business	7

Question 3	
Task	For all customers, find the number of policies they hold.
Explanation	This query should return two columns, one for the CustomerID, and one for the number of policies they have purchased.
SQL Solution	`SELECT C.CustomerID, COUNT(*) AS NumberOfPolociesOwned FROM Customer C JOIN Policy P ON P.CustomerID = C.CustomerID GROUP BY C.CustomerID;`
Output Screenshot	

```sql
SELECT C.CustomerID, COUNT(*) AS NumberOfPolociesOwned

FROM Customer C

JOIN Policy P ON P.CustomerID = C.CustomerID

GROUP BY C.CustomerID;
```

CustomerID	NumberOfPolociesOwned
1	1
2	4
3	10
4	2
5	6
6	6
7	2
8	1
10	1
11	1
12	1
13	1
14	1
15	11

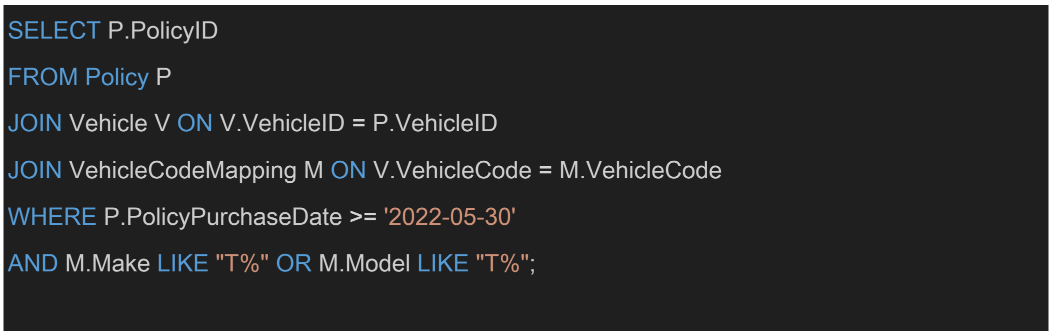
Question 4

Task

Policies purchased from 30 June 2022 onwards were given a discount if the Make or Model of any of their vehicles began with the letter ‘T’.

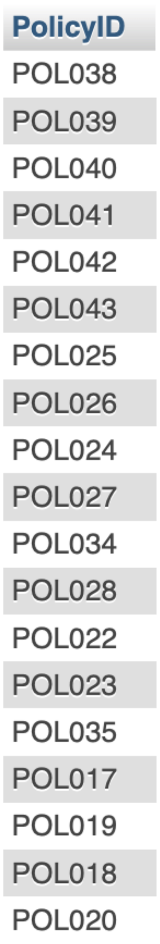
Return the PolicyID for all policies which received this discount.

SQL Solution



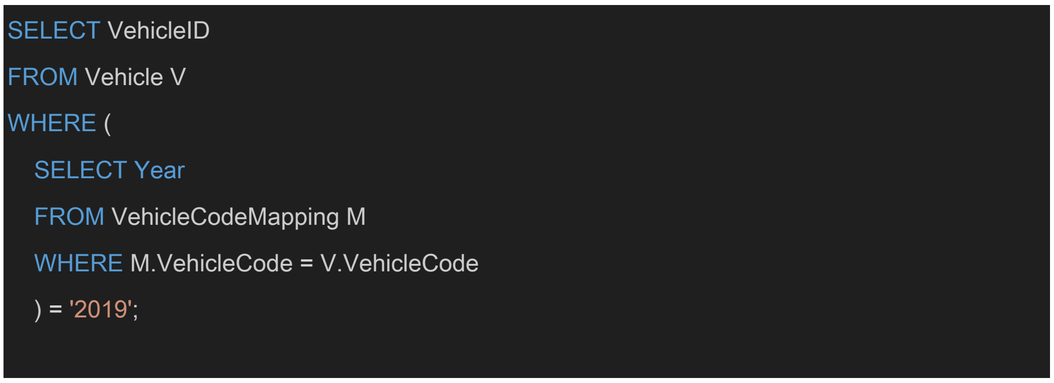
```
SELECT P.PolicyID
FROM Policy P
JOIN Vehicle V ON V.VehicleID = P.VehicleID
JOIN VehicleCodeMapping M ON V.VehicleCode = M.VehicleCode
WHERE P.PolicyPurchaseDate >= '2022-05-30'
AND M.Make LIKE "T%" OR M.Model LIKE "T%";
```

Output Screenshot



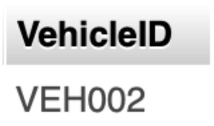
```
PolicyID
POL038
POL039
POL040
POL041
POL042
POL043
POL025
POL026
POL024
POL027
POL034
POL028
POL022
POL023
POL035
POL017
POL019
POL018
POL020
```

Question 5	
Task	Return the VehicleID of all vehicles that were made in the year 2019. Restriction: Use a sub-query to answer this question.

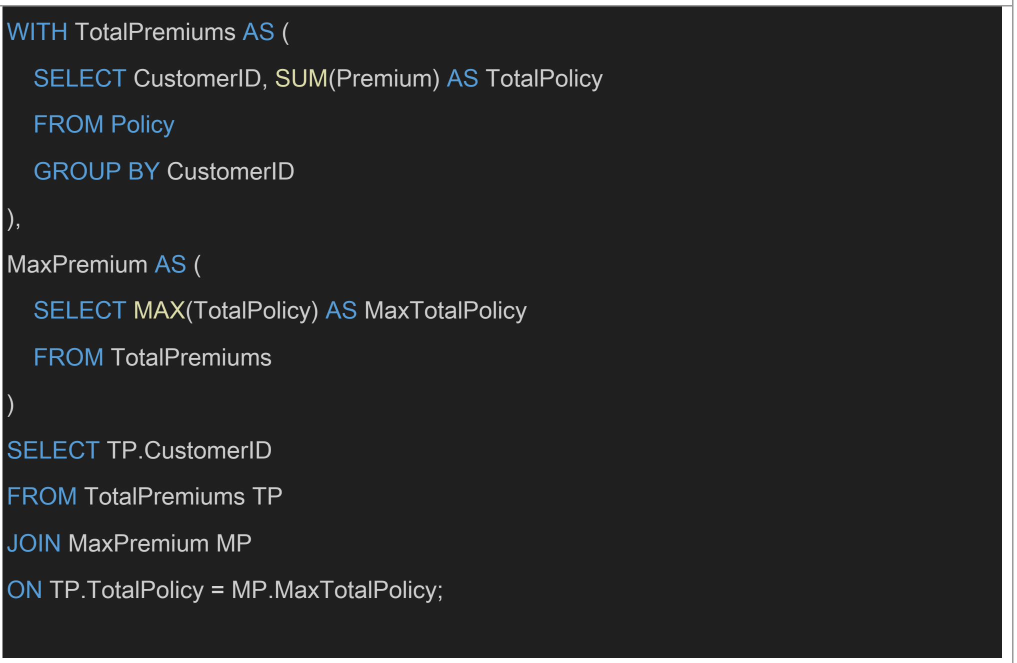
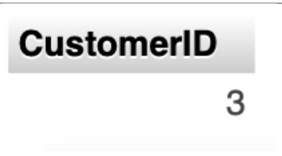


```
SELECT VehicleID
FROM Vehicle V
WHERE (
   SELECT Year
   FROM VehicleCodeMapping M
   WHERE M.VehicleCode = V.VehicleCode
   ) = '2019';
```

Output Screenshot



```
VehicleID
VEH002
```

Question 6	
Task	Return the id of customer(s) that have paid the highest premiums across all their policies compared to other customers.
Explanation	For example, if customer A has purchased four policies totalling $4000 in premiums, and customer B has purchased two policies totalling $5000 in premiums, the ID of customer B should be returned.
SQL Solution	
Output Screenshot	

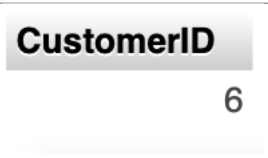
```sql
WITH TotalPremiums AS (
   SELECT CustomerID, SUM(Premium) AS TotalPolicy
   FROM Policy
   GROUP BY CustomerID
),
MaxPremium AS (
   SELECT MAX(TotalPolicy) AS MaxTotalPolicy
   FROM TotalPremiums
)
SELECT TP.CustomerID
FROM TotalPremiums TP
JOIN MaxPremium MP
ON TP.TotalPolicy = MP.MaxTotalPolicy;
```

```
CustomerID
         3
```

Question 7	
Task	Find and return the customerID of any customers who have insured at least all Tesla models captured in the VehicleCodeMapping table.
Explanation	Tesla is a vehicle make that includes models like the Model S and Model 3.

```sql
SELECT INSURED.CustomerID
FROM (
   SELECT p.CustomerID, COUNT(DISTINCT vm.VehicleCode) AS InsuredTeslaModels
   FROM Policy p
   JOIN Vehicle v ON p.VehicleID = v.VehicleID
   JOIN VehicleCodeMapping vm ON v.VehicleCode = vm.VehicleCode
   WHERE vm.Make = 'Tesla'
   GROUP BY p.CustomerID
) INSURED
JOIN (
   SELECT COUNT(DISTINCT vm.VehicleCode) AS TotalTeslaModels
   FROM VehicleCodeMapping vm
   WHERE vm.Make = 'Tesla'
) AVAILABLE
ON INSURED.InsuredTeslaModels = AVAILABLE.TotalTeslaModels;
```

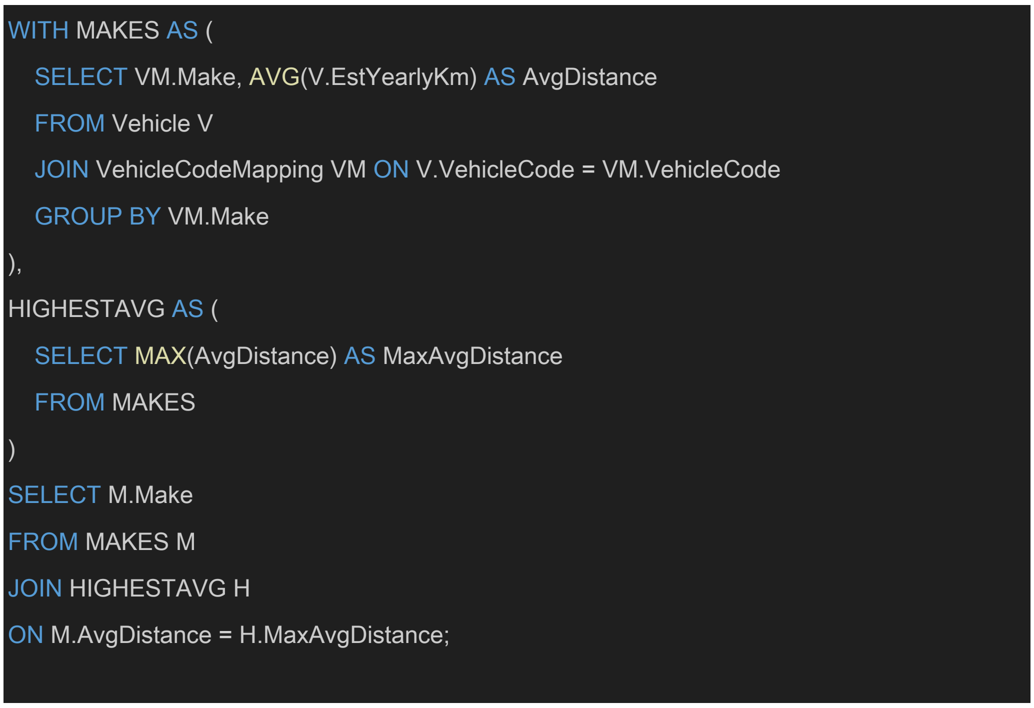
Output Screenshot



CustomerID
6

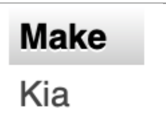
Question 8	
Task	Find the vehicle make(s) with the highest average estimated yearly kilometres.
Explanation	**Hint**. You may want to use one or more views in your answer.

SQL Solution



```
WITH MAKES AS (
   SELECT VM.Make, AVG(V.EstYearlyKm) AS AvgDistance
   FROM Vehicle V
   JOIN VehicleCodeMapping VM ON V.VehicleCode = VM.VehicleCode
   GROUP BY VM.Make
),
HIGHESTAVG AS (
   SELECT MAX(AvgDistance) AS MaxAvgDistance
   FROM MAKES
)
SELECT M.Make
FROM MAKES M
JOIN HIGHESTAVG H
ON M.AvgDistance = H.MaxAvgDistance;
```

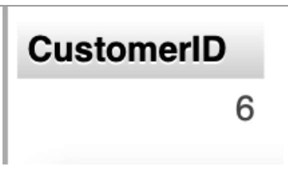
Output Screenshot



```
Make
Kia
```

Question 9	
Task	"Business class" customers are customers who have policies for 3 or more different vehicles that are for Business purposes (and any number of other vehicles). "UQ associate" customers are customers who have an email ending with uqconnect.edu.au or uq.edu.au. Find the customer ID of all customers who are both "Business class" and "UQ associates." **Restriction:** You must use a **set operation** in your answer.
Explanation	**Hint**. You may want to use one or more views in your answer.
SQL Solution	

```sql
SELECT C.CustomerID
FROM Customer C
JOIN (
    SELECT P.CustomerID
    FROM Policy P
    JOIN Vehicle V ON P.VehicleID = V.VehicleID
    WHERE V.VehiclePurpose = 'Business'
    GROUP BY P.CustomerID
    HAVING COUNT(P.VehicleID) >= 3
) BC ON C.CustomerID = BC.CustomerID
WHERE C.Email LIKE '%uqconnect.edu.au%'
  OR C.Email LIKE '%uq.edu.au%';
```

Output Screenshot	 CustomerID 6

Section B – SQL DML (UPDATE, DELETE, INSERT)

Question 1	
Task	Delete all policies held by Elena Ivanova that were purchased before December 30, 2023.

SQL Solution	`DELETE  P`
	`FROM Policy P`
	`JOIN Customer C ON  P.CustomerID = C.CustomerID`
	`WHERE C.Name = 'Elena Ivanova'`
	`AND  P.PolicyPurchaseDate < '2023-12-30';`

Question 2	
Task	A new fleet of vehicles organised by a customer called James Moran have had their VehiclePurpose incorrectly entered as ‘Private’. Update all of James’ vehicles to be insured as ‘Business’ instead.

SQL Solution	
	```
	UPDATE Vehicle V
	JOIN Policy P ON V.VehicleID = P.VehicleID
	JOIN Customer C ON P.CustomerID = C.CustomerID
	SET V.VehiclePurpose = 'Business'
	WHERE C.Name = 'James Moran'
	  AND V.VehiclePurpose = 'Private';
	```

Section C – SQL DDL

Question 1

Task

Create a new relation named **InsuranceClaim** to capture details of claims submitted by customers against their insurance policies. This table is designed to store comprehensive information about each claim, including the incident date, the date the claim was filed, the claimed amount, and the current status of the claim.

EasyDrive Insurance intends to capture the following details in their new relation:

- **ClaimID**: A unique identifier for each claim, automatically incremented.
- **PolicyID**: A foreign key linking to the Policy table, indicating the specific policy under which the claim is made.
- **ClaimDate**: The date on which the incident leading to the claim occurred.
- **ClaimAmount**: The monetary amount requested by the customer in the claim.
- **ClaimStatus**: The current status of the claim ('Pending', 'Approved', 'Rejected').
- **ClaimDescription**: A brief narrative providing details about the incident and the claim.

Write a SQL DDL query to implement the relation **InsuranceClaim.**

SQL Solution

```
CREATE TABLE InsuranceClaim (
    ClaimID INT AUTO_INCREMENT PRIMARY KEY,
    PolicyID VARCHAR(6),
    ClaimDate DATE NOT NULL,
    ClaimAmount DECIMAL(10, 2) NOT NULL,
    ClaimStatus VARCHAR(10),
    ClaimDescription VARCHAR(500),
    CONSTRAINT FK_Policy_Claim FOREIGN KEY (PolicyID) REFERENCES Policy(PolicyID)
);
```

Question 2

Task

Add a constraint to ensure that the estimated yearly kilometres for any vehicle is at least 5,000 km per year.

Explanation

The following resources may be useful when answering this question:
Check constraints

SQL Solution

```
ALTER TABLE Vehicle
ADD CONSTRAINT chk_EstYearlyKm
CHECK (EstYearlyKm >= 5000);
```

Section D – Critical Thinking

In this section, you will receive theoretical situations related to the UoD mentioned in the task description. Your task is to offer strategies to tackle the situation and write SQL queries to execute the approaches.

Question 1	
Task	In the upcoming 2025 financial period, EasyDrive Insurance no longer wishes to insure customers they deem 'risky.' Based on the given schema, first propose three strategies for identifying "risky customers." And then Write SQL code to implement one of these strategies.
Explanation	You can use the relation from Question C.1 to answer this question.
Three Strategies	
SQL Solution	

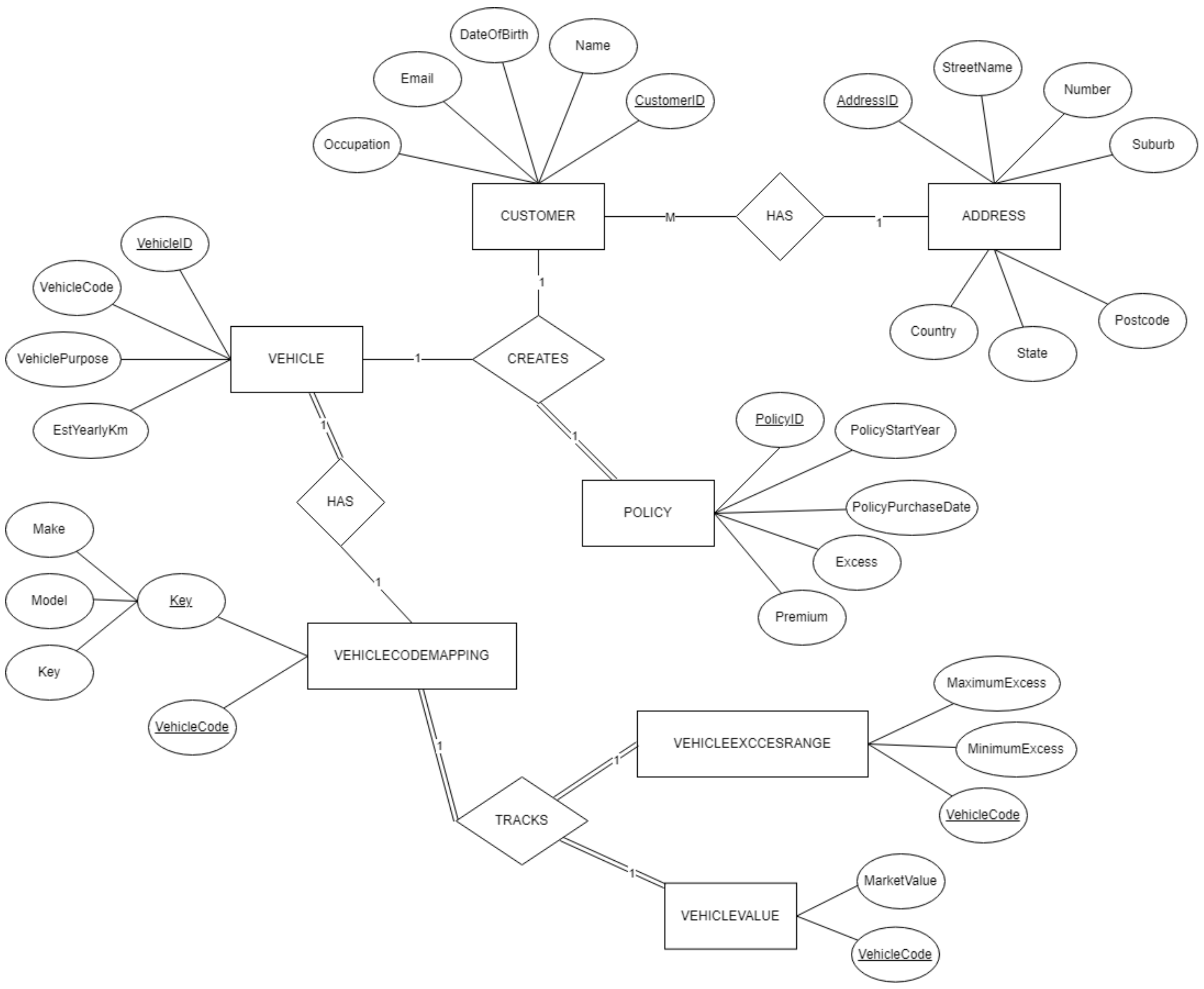
Question 2 – INFS7900 only	

Task	You are a senior DB admin managing a new graduate software engineer in your team. You have set them task of writing SQL for the following query:
	
	“Retrieve the names and emails of customers, along with the make, model, year, policy start year, premium, and the market value of their vehicles. Only include vehicles whose make starts with 'M' or 'N'. Include customers even if they don't have any associated vehicle or policy.”
	
	The new graduate has produced the following code:
	
	`SELECT`
	`    c.Name AS CustomerName,`
	`    c.Email AS CustomerEmail,`
	`    vcm.Make AS VehicleMake,`
	`    vcm.Model AS VehicleModel,`
	`    vcm.Year AS VehicleYear,`
	`    p.PolicyStartYear,`
	`    p.Premium,`
	`    vv.MarketValue`
	`FROM`
	`    Customer c`
	`LEFT JOIN`
	`    Policy p ON c.CustomerID = p.CustomerID`
	`JOIN`
	`    Vehicle v ON p.VehicleID = v.VehicleID`
	`JOIN`
	`    VehicleCodeMapping vcm ON v.VehicleCode = vcm.VehicleCode`
	`JOIN`
	`    VehicleValue vv ON v.VehicleCode = vv.VehicleCode`
	`WHERE`
	`   1=1`
	`   AND vcm.Make = 'Mitsubishi'`
	`   OR vcm.Make = 'Nissan';`
	
	There are errors in the code (more than one). In the three boxes below:
	i) identify and explain the errors,
	ii) describe alternative correct strategies for running the query, and
	iii) provide a new SQL statement correctly implementing the query.

Errors in the code	

Alternative strategies	
SQL Solution	

Appendix 1 – ER diagram for *EasyDrive Insurance*



DateOfBirth
Name
Email
CustomerID
Occupation
StreetName
Number
AddressID
Suburb
CUSTOMER
M
HAS
1
ADDRESS
VehicleID
VehicleCode
1
Country
Postcode
State
VehiclePurpose
VEHICLE
1
CREATES
EstYearlyKm
1
1
PolicyID
PolicyStartYear
HAS
POLICY
PolicyPurchaseDate
Make
Excess
1
Model
Key
Premium
Key
VEHICLECODEMAPPING
VehicleCode
MaximumExcess
1
1
VEHICLEEXCCESRANGE
MinimumExcess
VehicleCode
TRACKS
1
MarketValue
VEHICLEVALUE
VehicleCode