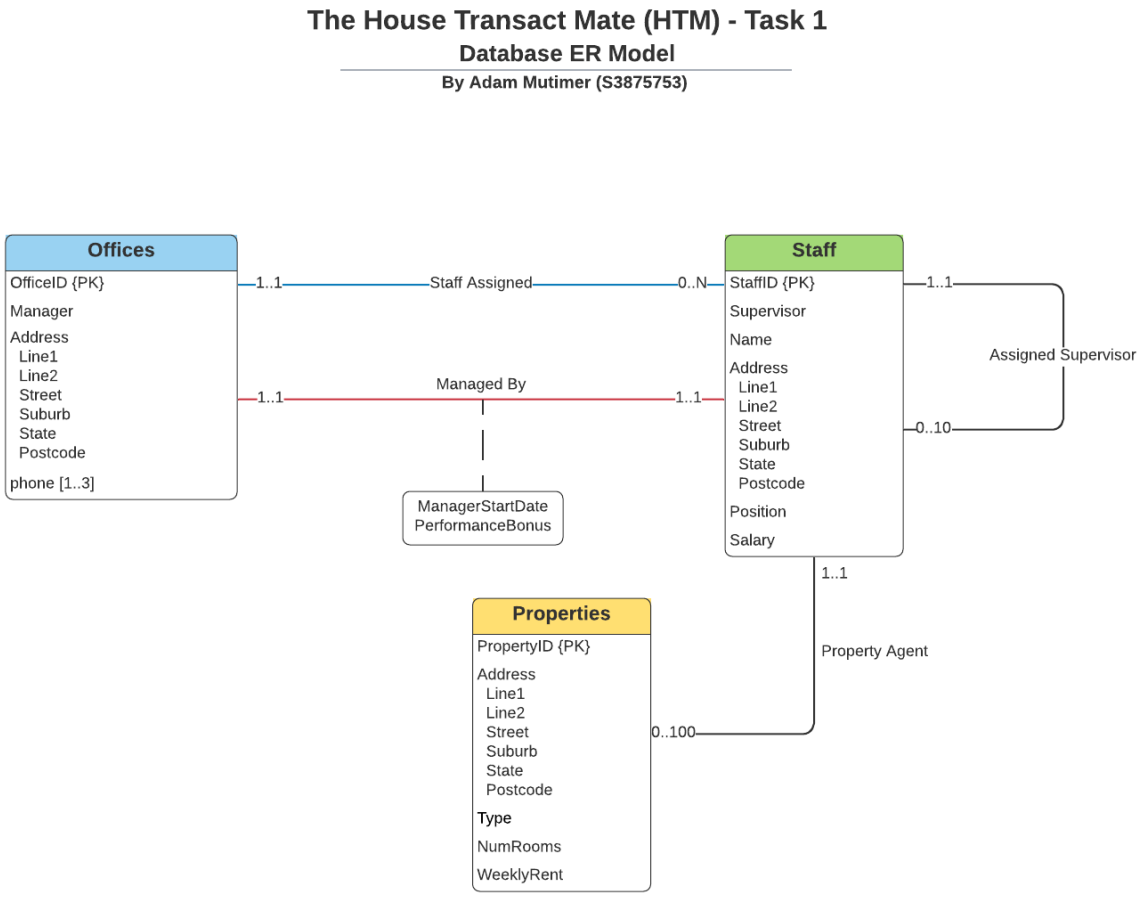
ISYS2095 – Assessment 1

Adam Mutimer (S3875753)

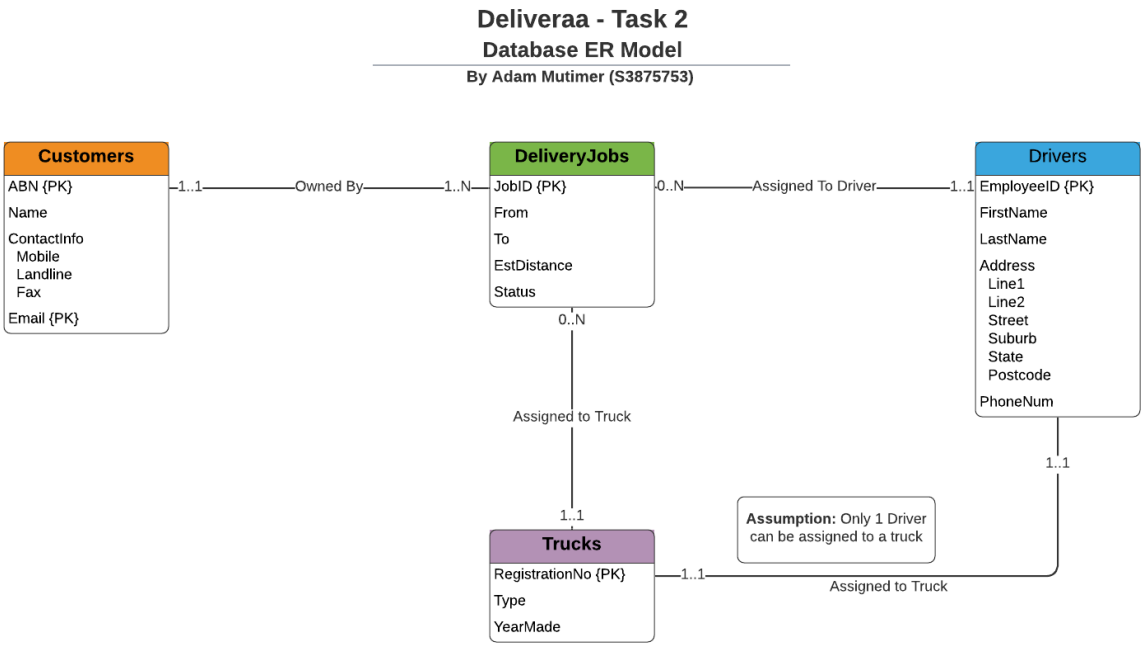
Task 1: Designing an Entity-Relationship Model



I believe my diagram clearly shows my assumptions for this task.

Task 2: Designing an Entity-Relationship Model

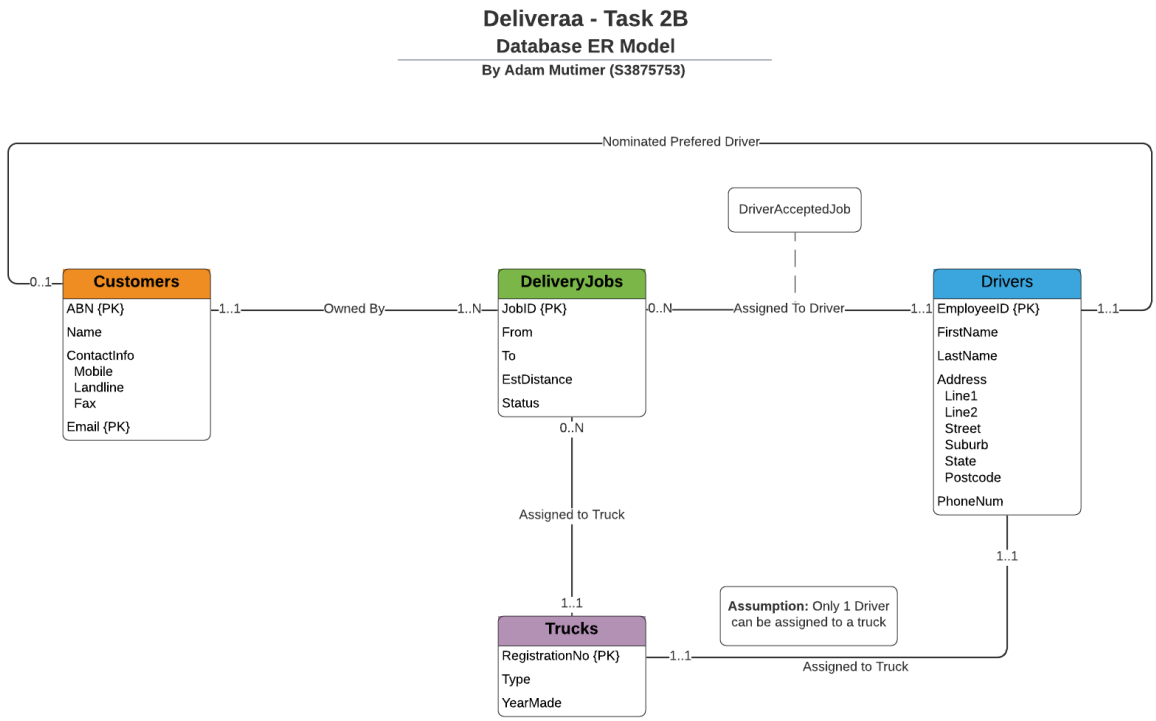
Part A)



For this task I believe my assumptions are clear; I have assumed that only one driver can be assigned to a truck as delivery jobs are assigned to both a driver and a truck.

Which is problematic if the delivery assigned to a driver and a truck, but the driver only has one delivery assigned to that truck whilst all his other deliveries are in another truck.

Part B)

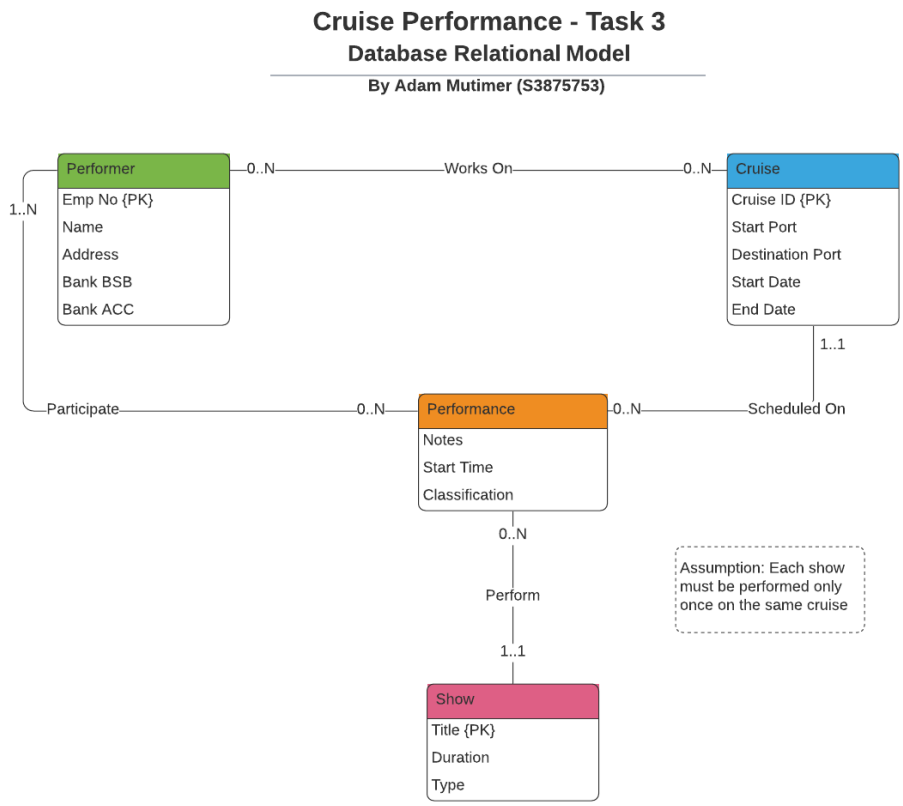


As per the clients change of specification, I have amended the ER model to reflect the new requirements.

- Customers are now able to be assigned a preferred driver (only 1)
- Drivers are now able to accept or reject jobs and a report can be generated using this modal.

All other assumptions remain the same as “Part A” at this time.

Task 3: Mapping the ER model to the relational model



Step 1: Map Entities

a) Map 1:1 relationship's

Performer (Emp No, Name, Address, Bank BSB, Bank ACC)

Cruise (Cruise ID, Start Port, Destination Port, Start Date, End Date)

Show (Title, Duration, Type)

b) Map weak entities

Performance (Show_Title, Notes, Start Time, Classification)
Together (Title, Start Time) form the primary key of the new relation (underlined)

Step 2: Map Relationships

1) Map 1:1 relationship's

Performer (Emp No, Name, Address, Bank BSB, Bank ACC)
Cruise (Cruise ID, Start Port, Destination Port, Start Date, End Date)
Show (Title, Duration, Type)
Performance (Notes, Start Time, Classification, Show_Title*, Cruise_ID*, ParticipateGroudID*)

2) *Map 1:n relationship's*

None

3) *Map N:M relationship's*

Works On (Emp_No*, CruiseID*)
Together Emp_No and CruiseID create a composite key

Participate (Emp_No*, ParticipateGroup)
Together Emp_No and ParticipateGroup create a composite key.

Step 3: Map multivalued attributes

No multivalued attributes

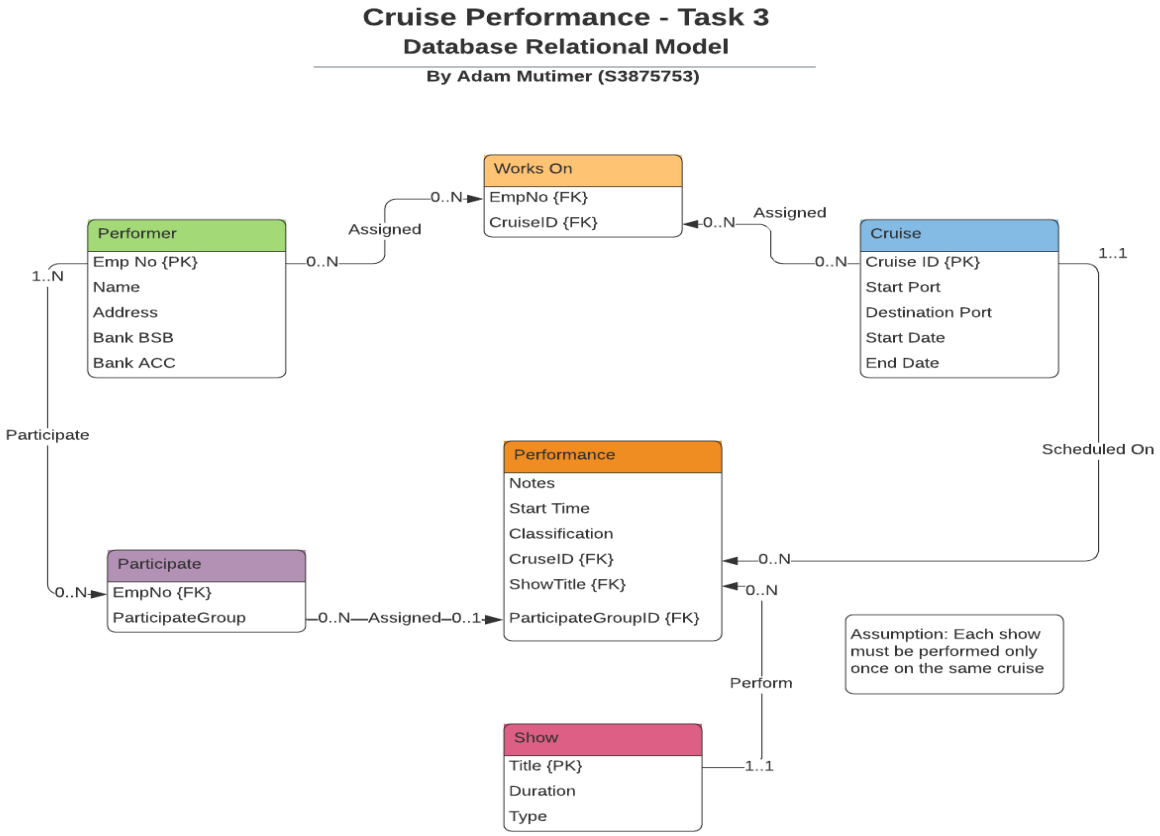
Step 4: Map higher degree relationships

No higher degree relationships

Step 5: Final Schema

Performer (Emp No, Name, Address, Bank BSB, Bank Acc)
Cruise (Cruise ID, Start Port, Destination Port, Start Date, End Date)
Performance (Notes, Start Time, Classification, CruseID*, ShowTitle*, ParticipoateGroupID*)
Show (Title, Duration, Type)
Works On (EmpNo*, CruiseID*)
Participate (EmpNo*, ParticipateGroup)

The below diagram I made while mapping out the diagram and isn't required by this task or assignment and is only included as I found it useful.



Task 4: Relational Database Model

1) **Does the database schema ensure that there is a job associated with each employee?**
   No, the database scheme dose not ensure that there will be a job associated with each employee, as **empjob_id** is not a foreign key, allowing any data to be entered depending on the data type of this column, the value could also be set to NULL assuming the column allows NULL values.

2) **Can an employee work for two departments at the same time? Can an employee take two jobs at the same time?**
   Technically yes but the data wouldn't be valid.
   as **department_id** is not a foreign key, the following query on **SQLLite** would technically allow it:

```
UPDATE Employees SET department_id = "1,2" WHERE employee_ID = 66;
```

   However, Most Database would not allow this if the column is an INTEGER

3) **The Human Resource department has recently changed to have three sub-departments (i.e., Ongoing Staff Department, Casuals Department, and External Contractors Department).**

Now, each sub-department is supposed to have a separate manager. Temporarily and until the new managers are hired, Joseph has been assigned to the management of all three sub-departments. The following SQL statements are intended to record all the changes required in the database instance. Will they work?

```
INSERT INTO Departments VALUES(3, 'Human Resource', 18, 30);
INSERT INTO Departments VALUES(3, 'Human Resource', 18, 30);
```

In this case I don't believe either of the two identical SQL Insert statements would be successful as the first column **department_id** is a primary key and would require the UNIQUE constraint on the field. So, the query would fail.

However hypothetically, if we ignore the unique constraint the primary key should have, then the query would add 2 entries with the same ID, Name, Manager and Location ID.

To accomplish the task according to the Database schema the queries should be as follows:

```
INSERT INTO Departments VALUES(4, 'HR: Casuals', 18, 30);
INSERT INTO Departments VALUES(5, 'HR: EXT.Contractors', 18, 30);
```

These two queries give the department a unique **department_id** and **name**, however the name is not required to be unique, and assigns the department manager to "Joseph Ryan" as well as setting the location to "555 Swanston St".

4) **The employee named Jonny Deans has recently been promoted to Senior Programmer. The following SQL statement intends to make the required changes in the database instance to reflect Adam's promotion.**

```
UPDATE Departments SET empjob_id=45 WHERE employee_id=10;
```

After running the above query, consider the request "find all the past contracts that Jonny Deans used to have". Can this request be completed using the given database schema and after the above statement is run? If yes, explain how the request can be answered. If no, explain what is missing and how it should be fixed.

No, the query would fail as there is now column **empjob_id** in the Departments table.
The correct method to accomplish this job change would be the following queries:

```
SELECT end_date FROM JobHistory WHERE employee_id = 10 ORDER BY end_date DESC LIMIT 1;
```

Would Return the end date of his previous position that we can use as the start date of his current position.
end_date = 20/08/2002

Now we can run this query to get his current **department_id** and **empjob_id**
SELECT department_id, empjob_id FROM Employees WHERE employee_id = 10 LIMIT 1;

Would Return:
department_id = 1
Empjob_id = 33

Now finally we can run the following query to update the JobHistory Table (Note we use current date as end_date.

INSERT into JobHistory Values (10, '29/08/2002', '01/07/2021', 33, 1);

Now finally we would update his Employees table record:
UPDATE Employees SET empjob_id=45 WHERE employee_id=10;

The following query could then be used to generate a report on his past contracts:
SELECT * FROM JobHistory where employee_id=10 ORDER BY start_date DESC;

5) **Explain what the result of executing the following SQL statement on the database instance will be.**

UPDATE LOCATIONS SET country_id=1,2 WHERE location_id=20;

The result would be a syntax error as it is not a valid SQL statement/query.
However, if the query/statement was formatted to be a valid query/statement as below, it would also fail due to the foreign key restraint as there is no matting entry in the **Countries** table.

UPDATE LOCATIONS SET country_id='1,2' WHERE location_id=20;

6) **Write an SQL statement to create the Jobs table including all the constraints. Make reasonable assumptions for the data type associated with each field. Your SQL statement must be valid for SQLite Studio environment and free of any errors.**

The following query would create the **Jobs** table including all the constraints:

CREATE TABLE Jobs (job_id INTEGER PRIMARY KEY, job_title STRING, min_salary STRING, max_salary STRING);

7) **Write an SQL statement to create the Employees table including all the constraints, assuming all the tables that Employees depends on already exist in the database. Make reasonable assumptions for the data type associated with each field. Your SQL statement must be valid for SQLite Studio environment and free of any errors.**

The following query would create the **Employees** Table including all the constraints:

CREATE TABLE Employees (employee_id INTEGER PRIMARY KEY, first_name STRING, last_name STRING, phone_number INTEGER, hire_date DATE, empjob_id INTEGER, salary STRING, department_id INTEGER);

8) **A new employee has been hired. You are asked to update the given database instance so that it also includes the new employee. The new employee's ID is 100, his name is "Aladdin Brown", his contract will start at 1 July 2021, and he works in the Human Resources department. Other information about Aladdin's appointment will be finalised later. Your SQL statement must be valid for SQLite Studio environment, free of any errors, and compatible with your answer to the previous question.**

The following query would update the Employees database to add the new employee "Aladdin Brown" with the supplied information in relation to his employment:

```
INSERT INTO Employees VALUES(100, 'Aladdin', 'Brown', NULL, '1/07/2021', NULL, NULL, 3);
```