Student ID: 730091091
ECM1420DA Exam

1.

a)

i. A full backup backs up a whole database. This includes part of the transaction log so that the full database can be recovered after a full database backup is restored. A differential backup is based on the most recent, previous full database backup. It captures only the data which has been changed since that full backup.

ii. Creating a differential backup can be very fast compared to creating a full backup. This is because the differential backup only records the data that has been changed since the last full backup.

iii. Differential backups make the restore process more complicated. This is because to recover the database, you first need to restore the latest backup, and then process the latest differential backup.

iv. First, restore HumanResourcesful72.bak. This is because it is the latest full backup. Then, restore HumanResourcesdiff346.bak. This is because it is the latest differential backup that was taken after the latest full backup. Next, process all of the transactions from HumanResourceslog1988.bak that occurred after the time 15/01/21 02:12 (The time that the latest differential backup was created). Finally, process all of the transactions in HumanResourceslog1989.bak to restore the database to its original state.

v. In the final step, only process the transactions that occurred before the time 17/01/21 04:00. This will prevent the restored database from being corrupted.

b) The size of the data that the programmer is trying to insert is larger than the maximum number of bits allocated for that field. This maximum size is assigned in the DDL script when the table is created.

c) There is a foreign key constraint for dptId and the programmer's data does not comply with this constraint. This is because the value that the programmer is trying to insert does not exist as one of the primary keys in the Department table.

d)

i. SELECT ItemName, CategoryName FROM StockItem, StockCategory WHERE StockItem.ItemCategoryId=StockCategory.CategoryId ORDER BY ItemName;

ii. An index is an on-disk structure associated with a table or view that speeds the retrieval of rows from the table or view. An index contains keys built from one or more columns in a table or view. In

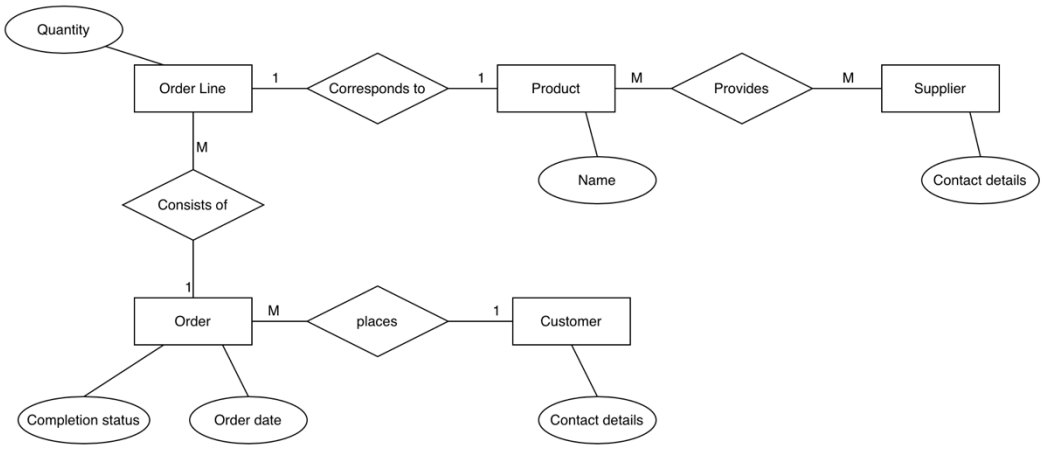
this case, you could have an index for ItemName and an index for CategoryName, which would speed up the execution of the query.

iii. SELECT CategoryName FROM StockCategory LEFT JOIN StockItem ON StockCategory.CategoryId=StockItem.ItemCategoryId WHERE ItemId IS NULL;

e) Firstly, I would create a view which SELECTs the date, time and location of all booked practice appointments. I would then create a login for each receptionist with their own password. This avoids the use of shared passwords and receptionists can be added when they join or removed when they leave the practice. I would then create a user for each login and grant CREATE SESSION to these users. Next, I would create a receptionist role and grant this role to all of the receptionist users. This role can then be revoked when certain receptions no longer require access to the data. Finally, I would grant SELECT on the newly created view to the receptionist role. As an additional feature, for ease of access and use, we can create stored procedures which SELECT a subset of the view and GRANT EXECUTE on these stored procedures to the receptionist role. This may include a procedure to SELECT all of the appointments, one to SELECT only today's appointments, one to SELECT all of the appointments at a specific location, etc. If certain receptionists are familiar with using SQL and would like to create their own stored procedures, we can GRANT CREATE PROCEDURE to the user so that they can create new procedures that save them time for certain tasks.

2.

a)



b) Module table – Module, Term Number, Module Fee
Student table – First name, Surname, Date of Birth
Enrolment – ModuleID, StudentID, Enrolment Date, Grade

3.

a)

i. There are various non-key and partial dependencies. For example, Agent is dependent on AgentId and ProductGroup is dependent on

ProductID. These should be segregated into their own tables if the data was to be normalised.

ii. When analysing data, it is much easier to use a non-normalised database. Using a normalised relational database to analyse data would require the creation of complex relation queries, which would take time and effort to formulate. It also discourages the use of ad-hoc queries. This kind of database where non-normalised data is stored for analysis is often known as a data warehouse. Using un-normalised data also allows for easier 'slicing and dicing' of the OLAP cube by filtering by dimensions.

iii. The total value sold by each agent can tell us who is performing well and allow us to set sales targets. Also, some agents may be better at selling certain products or product types, allowing for potential specialisation in these areas to increase sales.
The most popular product groups by region can allow us to target the sale of certain types of vehicles in specific regions, allowing us to increase the choice of desired types of vehicles available to customers in certain regions, potentially increasing sales.

iv. Dashboards are the best way to share views and data with colleagues. It will act as a centralised hub for the Sales Director, as well as other colleagues, to view the data and carry out BI analysis. The use of a dashboard may generate more questions about the data and lead to further investigations. The dashboard should contain a variety of charts, such as bar charts, line graphs, and pie charts. It should also contain toggles that allow filtering of the data, dynamically manipulating the charts to comply with this filtering. If desired, raw data tables may be also be included. The charts should be structured in a way that makes gaining business insights through analysis easy. Examples may include analysis of sales by agents, sales by quarter/year, sales by region, etc.

b)

i. A relational database ensures that data remains consistent and accurate. By centralising information into one system, you reduce the risk of data issues and conflicting records. The database enforces rules (such as unique keys and constraints) to maintain data integrity, preventing duplicate entries or invalid data.
In a relational database, data is organised into tables with relationships defined by keys. This structure minimises data redundancy since information is stored only once. This in turn optimises storage space and changes only need to be made in one place.

ii. The supplier code can be removed since site 2 and site 3 use different conventions, and site 1 did not use supplier codes. This will be replaced by a consistent SupplierID. Similarly, the Item ID/Item Code can be removed for the same reason. Redundant data also needs to be removed/combined. For example, there are two records for “Coach Screw 100mm” in Site 2.
iii. The Last delivered data format needs to be made consistent, or there will be formatting issues when inserting/transferring the data into the new database.
iv. LastDelivered DATE – makes sure the date is consistently formatted and that only valid dates can be entered. It also allows for filtering by month/year.
ItemID INT IDENTITY PRIMARY KEY – Adds a primary key constraint to make sure there is no data redundancy. Where a value is not specified when inserting data, a unique ID will be assigned.