CSG1207/CSI5135 Systems and Database Design

Assignment 2:	Database Design & Implementation (Pizza Store)
Assignment Marks:	Marked out of 60, (30% of unit) Task 1 is marked out of 20, (10% of unit) Task 2 is marked out of 40, (20% of unit)
Due Dates:	Inform tutor if working in a pair: 20 April 2015, 9:00AM Task 1 (Database Design): 27 April 2015, 9:00AM Task 2 (Implementation): 1 June 2015, 9:00AM

General Assignment Information

Before a database can be implemented and used, it must be designed in a way that ensures it is appropriate for the task at hand. Tools such as Entity-Relationship Diagrams and Data Dictionaries assist in designing and communicating the structure of a database. Once a design has been finalised, the database can be implemented in a DBMS. The Data Definition Language commands of SQL are used to create the database and its tables, attributes and constraints, after which the Data Manipulation Language commands can be used to manipulate data within the database. This assignment takes you through the design and implementation process, using Microsoft SQL Server.

You may work in **pairs** (maximum of **2** people) to complete this assignment, or choose to work **alone**. If you wish to work in a pair, you **must** inform your tutor (by email) of the names and student numbers of both members at *least one week before the due date of Task 1*. If you choose to work alone, be aware that there are *no extra marks* to compensate for the heavier workload. If working in a pair, I recommend *working through the whole assignment together*, rather than dividing it and completing sections individually. This helps to ensure that both people learn the content.

The assignment consists of two tasks. The first task, **Database Design**, requires a word processed document in PDF format detailing the design of your database. The second task, **Implementation**, is a collection of SQL scripts which create and populate the database designed in the first task, and then manipulate the data it contains. A small amount of marks are dedicated to **presentation, notation and formatting**.

While Task 1 is worth relatively few marks it is the *basis of Task 2*, and therefore it is very important that it be done well, otherwise further work will be required before Task 2 can be done.

Task 1 – Database Design (20 marks)

Your first task is to design a database for the scenario detailed on the following pages. Your final database design should include approximately 8 entities.

***Note:** The scenario for this assignment is the same as the one from Task 3/4 of Assignment 1. Be sure to incorporate any feedback received in Assignment 1.*

State any **assumptions** (2 marks) you have made regarding your database design at the beginning of the database design document. Do not make any assumptions that significantly change the structure of the scenario, as this may make Task 2 of the assignment difficult. Only make assumptions that influence your database design. If you are unsure about an assumption you wish to make, ask your tutor.

Once you feel you have identified the entities, attributes and relationships of the scenario in sufficient depth, you are required to create a **logical ER diagram** (4 marks) and a corresponding **physical ER diagram** (5 marks) to depict your database. Adhere to the distinctions between logical and physical ER diagrams covered in Lecture 3. It is recommended that you draw your diagrams on paper first, in order to find a layout that is clear and can be created in an electronic format.

Lastly, create a **data dictionary** (7 marks), with an entry for each entity in your database. The entries should list the name of the entity, a description of its purpose, a list of attributes (columns), important information about the attributes (e.g. data type, null/not null, identity, default values…), and details of any constraints applied to attributes. List the entries in your data dictionary in an appropriate *table creation order* that can be used to create the database. Include any additional information, if any, that may be needed to implement the database. Remember, a data dictionary should contain *all the information needed* to implement a database. Use the data dictionary in Lecture 4 as an example. Some marks are also awarded for **presentation and notation** (2 marks).

Your complete database design should consist of a list of assumptions, logical and physical ER diagrams and a data dictionary. This should be in the form of a *single PDF document*. An assignment cover sheet is also not required, but ensure that the first page of your assignment includes the unit code, assignment number/name, year and semester, your name and student number, your tutor's name, and the time and campus of your workshop session.

Please ensure that your completed assignment is in PDF format, and open the PDF file before submitting it to ensure that your diagrams appear as intended.

Scenario Details

You are required to design and create a database for a pizza store. The database must encompass the customers, staff, pizza details, and the pizza orders made by customers. You have the following information about the way the store operates:

- Customer details must be recorded. This includes a customer ID number, name, address and email. Customer details are recorded when they make their first order.

- Staff details must be recorded. This includes a staff ID number, first name, last name, date of birth and phone number.
  - Each staff member may have a supervisor, which is another staff member. A staff member may supervise many other staff members. Not all staff have a supervisor.

- The details of pizza orders must be recorded. This includes an order ID number, the date and time that the order was placed, the ID number of the customer who made the order, and the ID number of the staff member who took the order.
  - The table also needs to contain the staff ID number of the staff who delivered the order. Since the pizza order will be recorded *before* the pizzas are delivered, this value will originally be empty.
  - Each order can contain multiple pizzas.

- The store has divided their pizza selection into "ranges" (e.g. "traditional" and "gourmet") to simplify pricing. All of the pizzas in a range have the same price.
  - The database must store an ID, name and price for each range.

- The details of the types of pizza available must be recorded. This includes a pizza ID number, the pizza's name, a description and a foreign key identifying which pizza range it is in.

- The database also needs two tables to store the details of crust types and sauce types that can be chosen when ordering a pizza. Some crust/sauce types attract a surcharge.
  - These tables require an ID number, name and surcharge (default of 0) column.
  - When ordering a pizza, a customer must choose which crust and sauce they want.

- The database must track which pizzas were ordered in which orders. This will involve:
  - An auto-incrementing ordered pizza ID number.
  - A foreign key identifying the order that this pizza is part of.
  - A foreign key identifying which pizza was chosen.
  - A foreign key identifying which crust was chosen.
  - A foreign key identifying which sauce was chosen.
  - A "ready" column containing a "Y" or "N" to indicate whether the pizza has been made and cooked yet (default of "N").

General Information and Guidelines

The information above describes all of the entities, attributes and relationships required in the database design. Some minor details, such as the cardinality of some relationships, have been omitted. It is up to you to make (*and state*) any assumptions you need in order to complete the database design. If you are uncertain about any part of the scenario described above, seek clarification from your tutor.

It is recommended that you **use auto-incrementing integers as primary keys for all tables**. Be sure to specify the most appropriate data type (and length, where applicable) for each attribute in your data dictionary. In particular, store pizza range prices and surcharge amounts using SMALLMONEY, and staff date of birth values using DATE.

Read the scenario details *several times* to ensure that your database design incorporates all the elements described. *If you desire feedback on your work in progress, send it to your tutor*.

CSI5135 Additional Requirements

If you are in CSI5135, the following additional requirements apply. If you are in CSG1207, you do not need to do this (but you are welcome to do so if you want).

Ensure that your database design (and subsequent implementation in Task 2) specifies the following conditions upon the columns indicated. The conditions are:

- Customer email addresses must contain a "@" symbol.
    - You are welcome to implement stricter or more realistic email formatting if desired.
- Staff date of birth values must be at least 16 years earlier than the current date.
    - i.e. Staff must be at least 16 years old.
- The name of each pizza, crust and sauce must be unique within their respective tables.

Specify these check constraints in your data dictionary in a way that is as close to the actual SQL code required to create them as possible.

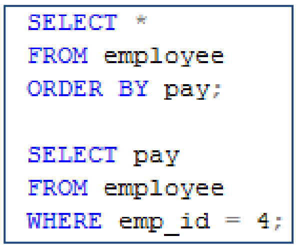
To enforce the conditions in your database you will need to use CHECK and UNIQUE constraints.

Task 2 – Implementation (40 marks)

Once your database has been designed, it is time to implement it in a DBMS, populate the database, and then manipulate the data via queries. The deliverables of this task are **three files containing SQL statements**. We will be using Microsoft SQL Server 2008 R2 – your SQL scripts *must* run in the same environment used in the unit/labs. If the database you implement is significantly different from your database design (Task 1), *include an updated database design document*, which notes the changes/additions to the design you submitted for Task 1.

Create your scripts as three ".sql" files, with the filenames listed in the following headings. **Templates for the script files are packages with this assignment brief** – please use them. Format your code for **readability**, and use **comments** if you wish to provide further detail or information about your code **(2 marks)**.

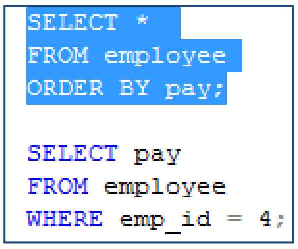
As each of the script files will contain numerous SQL statements, it is very useful to be aware of a particular feature of SQL Server Management Studio (SSMS): If you have selected some text in a query window, *only the selected text will be executed when you press the Execute button*.



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

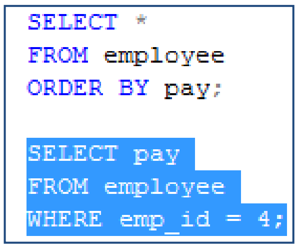
Execute both queries



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

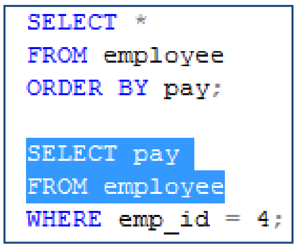
Execute query 1 only



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

Execute query 2 only



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

Execute part of query 2

This makes it much easier to test a single statement, or even part of a statement, within a script file.

You are required to create all of the scripts detailed below. Please take note of the file names and **use the template files downloaded with this assignment brief**.

Filename: create.sql

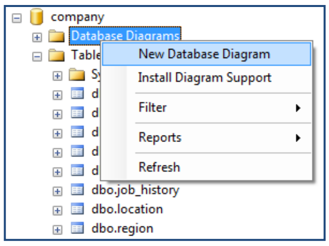
Database Creation & Population Script **(8 marks)**

Produce a script to create the database you designed in Task 1 (incorporating any changes you have made since then). Be sure to give your columns the same data types, properties and constraints specified in your data dictionary, and be sure to use a consistent naming scheme. Include any logical and correct default values and any check or unique constraints that you feel are appropriate.

Make sure this script can be run *multiple times* without resulting in any errors (hint: drop the database if it exists before trying to create it). Examine the creation scripts of the sample databases available in the unit materials for an example of how to do this.

You will need to follow an appropriate creation order when creating your tables – you cannot create a table with a foreign key constraint that refers to a table which does not yet exist.

Once you have created your database, it is recommended that you use SSMS to create an ER diagram and use this to verify that your implementation matches your design. This can be done by right clicking on the "Database Diagrams" folder of the database in the Object Explorer in SSMS.



Following the SQL statements to create your database and its tables, you must include statements to *populate the database with sufficient test data*. You are only required to populate the database with enough data to make sure that *all views and queries return meaningful results*. You may wish to start working on your views and queries and write INSERT statements that add the data needed to test each one as you go.

For example, imagine you are working on a query which draws data from two tables and only displays the rows that meet certain criteria, then orders the results by a certain column. To test this, you will need to insert some data in both of the tables, making sure that some of the rows meet the criteria and others don't, and making sure that the column used for ordering contains a range of different values. Once you have inserted this data, you can test that your query works.

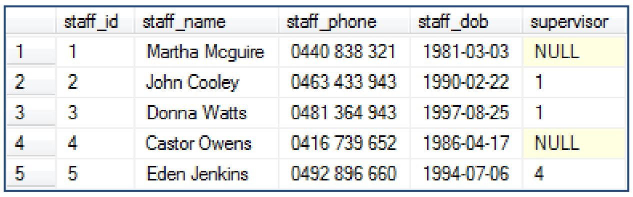
To make this task less time-consuming, ***I have provided all necessary data for the pizza range, pizza, crust and sauce tables in the "create_TEMPLATE.sql" file***. You will need to add your own data to the rest of the tables. The final create.sql should be able to create your database and populate it with enough data to make sure that all views and queries return meaningful results.

Make sure all referential integrity is observed – you cannot add data to a column with a foreign key constraint if you do not yet have data in the table it refers to. Remember that if you are using an auto-incrementing integer, *you cannot specify a value for that column when inserting a row of data*. Omit the column or specify a value DEFAULT, and the server will generate the number.

Filename: views.sql

Staff View **(1 mark)**

Create a *view* which shows staff ID, date of birth, phone number and supervisor ID of all staff members, as well as their first name and last name concatenated into a "full name" column.



	staff_id	staff_name	staff_phone	staff_dob	supervisor
1	1	Martha Mcguire	0440 838 321	1981-03-03	NULL
2	2	John Cooley	0463 433 943	1990-02-22	1
3	3	Donna Watts	0481 364 943	1997-08-25	1
4	4	Castor Owens	0416 739 652	1986-04-17	NULL
5	5	Eden Jenkins	0492 896 660	1994-07-06	4

(example of first five rows of view output – your data may vary, example illustrates structure only)

While very simple, this view can be used instead of the staff table in queries which require the full name of staff members so that you do not need to concatenate the first and last name every time.

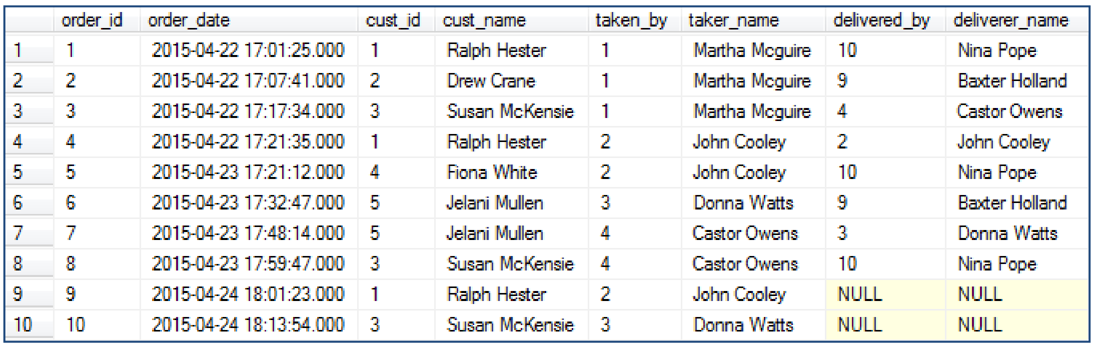
Pizza Orders View **(2 marks)**

Create a *view* which shows the following details of all rows in the "pizza order" table:

- The order ID number and order date.
- The customer ID number and name of the customer who placed the order.
- The staff ID number and full name of the staff member who *took* the order.
- The staff ID number and full name of the staff member who *delivered* the order.
    - Ensure that all orders are shown, even those which have not been delivered yet.

Using the Staff View in the query of this view is recommended.

Hint: *You will need two JOINs to the staff view/table, one of which needs to be an OUTER JOIN.*



	order_id	order_date	cust_id	cust_name	taken_by	taker_name	delivered_by	deliverer_name
1	1	2015-04-22 17:01:25.000	1	Ralph Hester	1	Martha Mcguire	10	Nina Pope
2	2	2015-04-22 17:07:41.000	2	Drew Crane	1	Martha Mcguire	9	Baxter Holland
3	3	2015-04-22 17:17:34.000	3	Susan McKensie	1	Martha Mcguire	4	Castor Owens
4	4	2015-04-22 17:21:35.000	1	Ralph Hester	2	John Cooley	2	John Cooley
5	5	2015-04-23 17:21:12.000	4	Fiona White	2	John Cooley	10	Nina Pope
6	6	2015-04-23 17:32:47.000	5	Jelani Mullen	3	Donna Watts	9	Baxter Holland
7	7	2015-04-23 17:48:14.000	5	Jelani Mullen	4	Castor Owens	3	Donna Watts
8	8	2015-04-23 17:59:47.000	3	Susan McKensie	4	Castor Owens	10	Nina Pope
9	9	2015-04-24 18:01:23.000	1	Ralph Hester	2	John Cooley	NULL	NULL
10	10	2015-04-24 18:13:54.000	3	Susan McKensie	3	Donna Watts	NULL	NULL

(example of view output – your data may vary, example illustrates structure only)

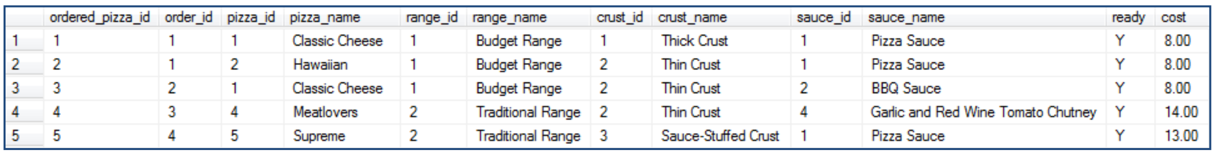
This view gives you a convenient way to view the names of the customer and staff members involved in each order, making queries that require this information more convenient to write.

Ordered Pizzas View (3 marks)

Create a *view* which shows the following details of all rows in the "ordered pizza" table:

- The ordered pizza ID number, order ID number and "ready" column.
- The pizza ID number and pizza name of the ordered pizza.
- The range ID number and range name of the ordered pizza.
- The crust ID number and crust name of the ordered pizza.
- The sauce ID number and sauce name of the ordered pizza.
- The cost of the pizza (add together the range price, crust surcharge and sauce surcharge)

Hint: *This requires four JOINs.*



	ordered_pizza_id	order_id	pizza_id	pizza_name	range_id	range_name	crust_id	crust_name	sauce_id	sauce_name	ready	cost
1	1	1	1	Classic Cheese	1	Budget Range	1	Thick Crust	1	Pizza Sauce	Y	8.00
2	2	1	2	Hawaiian	1	Budget Range	2	Thin Crust	1	Pizza Sauce	Y	8.00
3	3	2	1	Classic Cheese	1	Budget Range	2	Thin Crust	2	BBQ Sauce	Y	8.00
4	4	3	4	Meatlovers	2	Traditional Range	2	Thin Crust	4	Garlic and Red Wine Tomato Chutney	Y	14.00
5	5	4	5	Supreme	2	Traditional Range	3	Sauce-Stuffed Crust	1	Pizza Sauce	Y	13.00

(example of first five rows of view output – your data may vary, example illustrates structure only)

This view gives you a convenient way to view the names of the pizza, range, crust and sauce of each ordered pizza, as well as calculating the price. It can make some queries more convenient to write.

When writing a view, it is easiest to write the SELECT statement first, and only add the CREATE VIEW statement to the beginning once you have confirmed that the SELECT statement is working correctly.

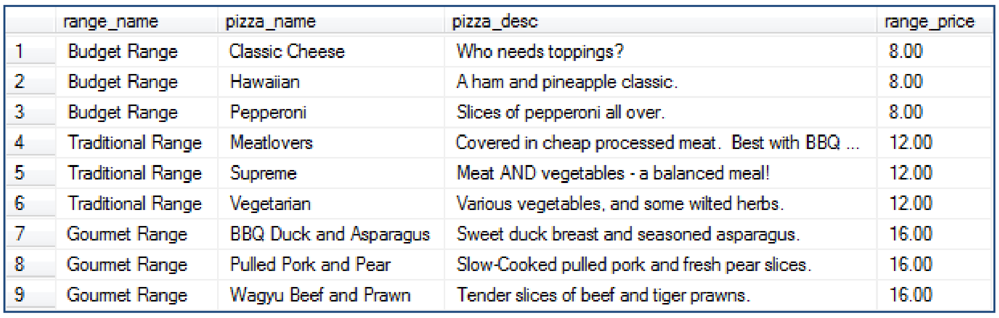
If you wish to create additional views to use in the queries which follow, include them in this file.

Filename: queries.sql

Write SELECT statements to complete the following queries. If you do not understand or are not sure about exactly what a query requires, contact your lecturer or tutor.

Query 1 – Pizza Menu (2 marks)

Write a query that shows the range name, pizza name, pizza description and range price of all pizzas. Order the results by range ID number, then pizza name.

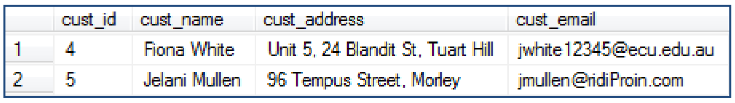


	range_name	pizza_name	pizza_desc	range_price
1	Budget Range	Classic Cheese	Who needs toppings?	8.00
2	Budget Range	Hawaiian	A ham and pineapple classic.	8.00
3	Budget Range	Pepperoni	Slices of pepperoni all over.	8.00
4	Traditional Range	Meatlovers	Covered in cheap processed meat. Best with BBQ ...	12.00
5	Traditional Range	Supreme	Meat AND vegetables - a balanced meal!	12.00
6	Traditional Range	Vegetarian	Various vegetables, and some wilted herbs.	12.00
7	Gourmet Range	BBQ Duck and Asparagus	Sweet duck breast and seasoned asparagus.	16.00
8	Gourmet Range	Pulled Pork and Pear	Slow-Cooked pulled pork and fresh pear slices.	16.00
9	Gourmet Range	Wagyu Beef and Prawn	Tender slices of beef and tiger prawns.	16.00

(example of query results)

Query 2 – Customer Search (2 marks)

Write a query that shows all details of customers with a name beginning with the letter "F" or "J", and whose email address is at least 15 characters long. Order the results by customer name.

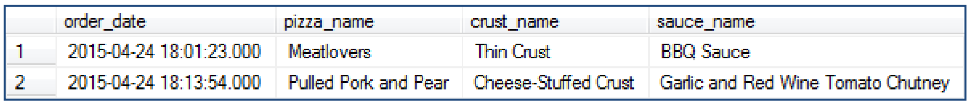


	cust_id	cust_name	cust_address	cust_email
1	4	Fiona White	Unit 5, 24 Blandit St, Tuart Hill	jwhite12345@ecu.edu.au
2	5	Jelani Mullen	96 Tempus Street, Morley	jmullen@ridiProin.com

(example of query results – your data may vary, example illustrates structure only)

Query 3 – Unready Pizzas (2.5 marks)

Write a query that shows the order date, pizza name, crust name and sauce name of any ordered pizzas which are not ready (ready column contains "N"). Order the results by the order date. Using the Ordered Pizzas View in this query is recommended.

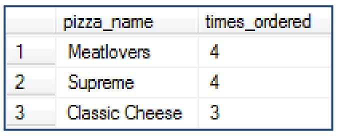


	order_date	pizza_name	crust_name	sauce_name
1	2015-04-24 18:01:23.000	Meatlovers	Thin Crust	BBQ Sauce
2	2015-04-24 18:13:54.000	Pulled Pork and Pear	Cheese-Stuffed Crust	Garlic and Red Wine Tomato Chutney

(example of query results – your data may vary, example illustrates structure only)

Query 4 – Popular Pizzas (3 marks)

Write a query that that shows the names of the top three most ordered pizzas and the number of times they have each been ordered. Using the Ordered Pizzas View in this query is recommended.



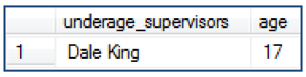
	pizza_name	times_ordered
1	Meatlovers	4
2	Supreme	4
3	Classic Cheese	3

(example of query results – your data may vary, example illustrates structure only)

Query 5 – Underage Supervisors (3 marks)

Write a query that shows the name and age (in years) of any staff members who younger than 21 years old and are supervising at least one other staff member. Using the Staff View in this query is recommended.

Hint: *Use the DATEDIFF function to determine the age of a staff member from their date of birth, and consider using a subquery and an IN comparison to determine if a staff member is supervising anyone.*



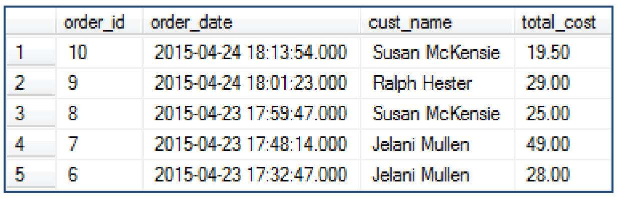
	underage_supervisors	age
1	Dale King	17

(example of query results – your data may vary, example illustrates structure only)

Query 6 – Pizza Order Summaries (3.5 marks)

Write a query that shows the order ID number, order date, customer name, and total cost of all pizza orders. Order the results by the order date, in descending order. Using the Pizza Orders View and the Ordered Pizzas View in this query is recommended.

Hint: *Use GROUP BY and SUM to group the results per order and calculate the total cost of the pizzas in each order.*



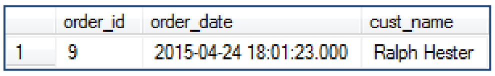
	order_id	order_date	cust_name	total_cost
1	10	2015-04-24 18:13:54.000	Susan McKensie	19.50
2	9	2015-04-24 18:01:23.000	Ralph Hester	29.00
3	8	2015-04-23 17:59:47.000	Susan McKensie	25.00
4	7	2015-04-23 17:48:14.000	Jelani Mullen	49.00
5	6	2015-04-23 17:32:47.000	Jelani Mullen	28.00

(example of first five rows of query results – your data may vary, example illustrates structure only)

Query 7 – Orders Awaiting Delivery (4 marks)

Write a query that shows the order ID number, order date and customer name of any orders which have not been delivered yet (delivery foreign key is NULL) and all pizzas in the order are ready (ready column contains "Y"). Order the results by the order date. Using the Pizza Orders View and the Ordered Pizzas View in this query is recommended.

Hint: *The WHERE clause which determines if the pizzas are ready and if the order has been delivered must come before the GROUP BY clause which groups the results per order.*



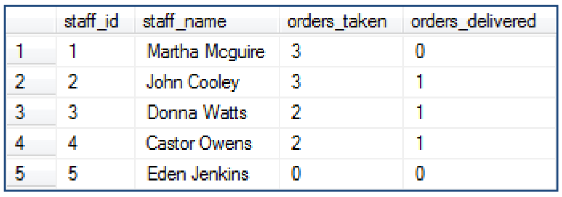
	order_id	order_date	cust_name
1	9	2015-04-24 18:01:23.000	Ralph Hester

(example of query results – your data may vary, example illustrates structure only)

Query 8 – Staff Workload (4 marks)

Write a query that shows the full name of all staff members, the number of orders they have taken and the number of orders they have delivered. Ensure that all staff members are included in the results, even those who have not taken or delivered any orders. Order the results by the staff ID number. Using the Staff View in this query is recommended.

Hint: *This query will require two OUTER JOINs, a GROUP BY clause and the COUNTing of DISTINCT order ID numbers.*



	staff_id	staff_name	orders_taken	orders_delivered
1	1	Martha Mcguire	3	0
2	2	John Cooley	3	1
3	3	Donna Watts	2	1
4	4	Castor Owens	2	1
5	5	Eden Jenkins	0	0

(example of first five rows of query results – your data may vary, example illustrates structure only)

Presentation, Notation and Formatting (2 marks per part)

A small amount of marks are awarded for presentation, notation and formatting. This includes:

- Presentation and appearance of word processed PDF document for Task 1
- Appropriateness and consistency of notation used for diagrams in Task 1
- Appropriate commenting and formatting of scripts in Task 2

Submission of Deliverables

Submit your database design (Task 1, a PDF document) by the Task 1 due date above. Submit your script files (Task 2) as single zipped file by the Task 2 due date above. Submit the files to the appropriate locations in the Assessments area of Blackboard.

If you are working in pairs, **both** people should submit the same assignment. Include the **name and student number of both members** in the first page of the assignment and submission comments field in Blackboard. Remember, if you wish to work in a pair, you **must** inform your tutor of the names and student numbers of both members at *least one week before the due date of Task 1*.

Submissions via email or hard copies are NOT permitted, unless you are specifically instructed to do so. An assignment cover sheet is not required, but ensure that the first page of your assignment includes the unit code, assignment number/name, year and semester, your name and student number, your tutor's name, and the time and campus of your workshop session.

Referencing & Plagiarism

The entirety of your assignment must be your own work, unless otherwise referenced, and produced for the current instance of the unit. Any use of unreferenced content you did not create constitutes plagiarism, and is deemed an act of academic misconduct. All assignments will be submitted to plagiarism checking software which includes previous copies of the assignment.

CSG1207/CSI5135 Systems and Database Design

Task 1 (Database Design) - Marks Allocation

Criteria	Marks
Assumptions All/Any assumptions that influence the database design clearly stated.	**2**
Logical ER Diagram Diagram accurately depicts the scenario and includes all elements specified in the brief.	**4**
Physical ER Diagram Accurately depicts the scenario and is a correct translation of the logical ER diagram.	**5**
Data Dictionary Includes all entities and details of attributes as specified in brief and correct creation order used.	**7**
Presentation and Notation Assignment is well presented, uses consistent and appropriate notation.	**2**

Total:	**20** **(10% of unit)**

Task 2 (Implementation) - Marks Allocation

Criteria All scripts are judged on correctness, appropriateness and readability of SQL code.	Marks
Database Creation & Population Script	**8**
Staff View	**1**
Pizza Orders View	**2**
Ordered Pizzas View	**3**
Query 1	**2**
Query 2	**2**
Query 3	**2.5**
Query 4	**3**
Query 5	**3**
Query 6	**3.5**
Query 7	**4**
Query 8	**4**
Formatting – Scripts are well formatted and use appropriate formatting.	**2**

Total:	**40** **(20% of unit)**