

**ECU178 Computer Science:
220CT Data and Information Retrieval
Coursework**

Due on March 13th 2015

Robert Rigler : 4939377

Contents

Task 1 Database Design	4
Activity 1: First Normal Form	4
Step 1: Identify Redundancy	4
Step 2: Create New Entity	4
Step 3: Relationships and Keys	4
1NF : Diagram	4
1NF : Data	5
Activity 2: Second Normal Form	6
Step 1: Testing each attribute.	6
Step 2: Entity, Relationship and Key	6
Entities in Second Normal Form: Diagram	6
Entities in Second Normal Form: With data	7
Activity 3: Third Normal Form	8
Step 1: Testing for Transitive Dependency	8
Step 2: Entity, Relationship and Key	8
Entities in Third Normal Form: Diagram	8
Entities in Third Normal Form: With Data	9
Activity 4: ER Diagram	10
Task 2: Database Development	10
Question 1	10
Aircraft	10
Airline	11
Purchase_Order	12
Ordered_Aircraft	13
Question 2	14
Aircraft	14
Airline	15
Purchase_Order	16
Ordered_Aircraft	17
Question 3	19
Question 4	20
Question 5	21
Question 6	22
Question 7	23
Question 8	24
Question 9	24
Task 3 : Poster of Ethics	25
Task 4 : Amazon recommendation system	26
What is a Recommendation System?	26
Data Gathering and Preparation	26
Data	26
Benefits and limitations of a Recommendation system	28
Limitations	28
Benefits	28
Analysis	30

Collaborative filtering	30
The Demographic approach	30
Bibliography	31
References for Task 3: A Poster of Ethics	31
References for Task 4: A Recommendation system	31

Task 1 Database Design

This task in Database Design consists of four activities. The first three involve normalising the given data to third normal form, and the fourth is to produce an Entity Relationship diagram of the normalised relations. For each activity I will give a detailed step by step explanation of how I completed each activity.

Activity 1: First Normal Form

To put this data into First Normal Form (1NF), I need to:

1. Identify any repeating redundant data and remove it from the current Entity
2. Place the data into a new Entity
3. Create a relationship with a primary key from one Entity as a foreign key in the other.

Step 1: Identify Redundancy

On inspecting the data, I can see that there are multiple instances of repeating data.

Orders' ID: *CON-2237*, *CON-2356* and *CON-1234* all have repeating data entries for fields: *Equipment*, *Qty*, and *Unit Price*.

Step 2: Create New Entity

Removing the *Equipment*, *Qty*, and *Unit Price* fields and placing them in a new entity, leaves me with two entities as shown below.

Step 3: Relationships and Keys

To complete the First Normal Form, a relationship needs to be created between the entities.

I created the relationship by including the *Order ID* attribute as a foreign key in the *ItemOrder* entity.

Order ID is used as a Primary Key for the *Order* entity. In the *ItemOrder* entity, no one attribute can be used to uniquely identify a single record. For this reason, I have created a concatenated key using the attributes *Equipment* and *Order ID*. The concatenated key can now be used to uniquely identify each record.

1NF : Diagram

Order	(<u>Order ID</u>	ItemOrder	(<u>*Order ID</u>
	Supplier ID		<u>Equipment</u>
	Client Name		Qty)
	Client Address		Unit Price)
	Date		
	Total Price)		

1NF : Data*Order entity*

Order ID	Supplier ID	Client Name	Client Address	Date	Total Price
CON-2237	168	Coventry Building Services Ltd	Units 2-4, Binley Industrial Estate, CV3 2WL	14-Dec-14	£99.00
CON-3664	527	Allied Construction Ltd	34, Lythalls La Industrial Estate, CV6 6RG	16-Jan-15	£36.00
CON-2356	169	Rioh Builds Ltd	Unit 12, Stoneleigh Park, CV8 2UV	12-Feb-15	£280.00
CON-1234	032	Grand Design Ltd	32-34, Bilton Industrial Estate, CV3 5YB	16-Apr-15	£23.00

ItemOrder entity

Order ID	Equipment	Qty	Unit Price
CON-2237	Butterfly Valve	2	£5.00
CON-2237	3/4" Locknut	6	£1.50
CON-2237	Sch 40 Blk Pipe	4	£20.00
CON-3664	Thin Stranded Copper Wire	6	£6.00
CON-2356	Sch 40 Blk Pipe	3	£20.00
CON-2356	4x8x3/4 Cos Plywood	2	£10.00
CON-2356	3/4" EMT	2	£50.00
CON-2356	Duplex Ivy Rec	1	£100.00
CON-1234	Sch 40 Blk Pipe	1	£20.00
CON-1234	3/4" Locknut	2	£1.50

Activity 2: Second Normal Form

To put this data into the Second Normal Form(2NF) I need to ensure that the attributes are completely dependant on the primary key, i.e., that no attribute is only dependant on one part of the primary key.

This can be done in two steps:

1. Test each attribute for complete dependency on the primary key.
2. Remove any partially dependent attributes to a new entity and assign a primary key.

For this particular set of data, the *Order* entity does not have a concatenated key and therefore is already in the second normal form.

Step 1: Testing each attribute.

Primary Key	Attribute	Functionally Dependant?
Order ID, Equipment	Qty	Yes, dependant on both
Order ID, Equipment	Unit Price	No, dependant on Equipment only

Step 2: Entity, Relationship and Key

From my testing, I found that the attribute *Unit Price* is not functionally dependant as it is only dependent on *Equipment*, but not *Order ID*.

I moved *Equipment* and *Unit Price* into a new entity called *Item* and made *Equipment* the primary key.

I created a relationship between the *Item* and *ItemOrder* entities by repeating the *Equipment* attribute in *ItemOrder* as a foreign key.

Entities in Second Normal Form: Diagram

Order	(<u>Order ID</u>	Item	(<u>Equipment</u>	ItemOrder	(* <u>Order ID</u>
Supplier ID		Unit Price)		* <u>Equipment</u>	
Client Name				Qty)	
Client Address					
Date					
Total Price)					

Entities in Second Normal Form: With data*Order entity*

Order ID	Supplier ID	Client Name	Client Address	Date	Total Price
CON-2237	168	Coventry Building Services Ltd	Units 2-4, Binley Industrial Estate, CV3 2WL	14-Dec-14	£99.00
CON-3664	527	Allied Construction Ltd	34, Lythalls La Industrial Estate, CV6 6RG	16-Jan-15	£36.00
CON-2356	169	Rioh Builds Ltd	Unit 12, Stoneleigh Park, CV8 2UV	12-Feb-15	£280.00
CON-1234	032	Grand Design Ltd	32-34, Bilton Industrial Estate, CV3 5YB	16-Apr-15	£23.00

Item entity

Equipment	Unit Price
3/4" EMT	50.00
3/4" Locknut	1.50
4x8x3/4 Cos Plywood	10.00
Butterfly Valve	5.00
Duplex Ivy Rec	100.00
Sch 40 Blk Pipe	20.00
Thin Stranded Copper Wire	6.00

ItemOrder entity

Order ID	Equipment	Qty	Unit Price
CON-2237	Butterfly Valve	2	£5.00
CON-2237	3/4" Locknut	6	£1.50
CON-2237	Sch 40 Blk Pipe	4	£20.00
CON-3664	Thin Stranded Copper Wire	6	£6.00
CON-2356	Sch 40 Blk Pipe	3	£20.00
CON-2356	4x8x3/4 Cos Plywood	2	£10.00
CON-2356	3/4" EMT	2	£50.00
CON-2356	Duplex Ivy Rec	1	£100.00
CON-1234	Sch 40 Blk Pipe	1	£20.00
CON-1234	3/4" Locknut	2	£1.50

Activity 3: Third Normal Form

To place the data into Third Normal Form 3NF I need to ensure that all attributes are only dependent on the Primary Key, and not Non-Key Attributes. This can be achieved in two steps:

1. Test each attribute for dependency on the primary key.
2. Remove all transitive dependencies to a new entity with the correct primary key and relationship.

For this particular set of data, both the *Item* and *ItemOrder* entities are already in the Third Normal Form.

Step 1: Testing for Transitive Dependency

Order Entity

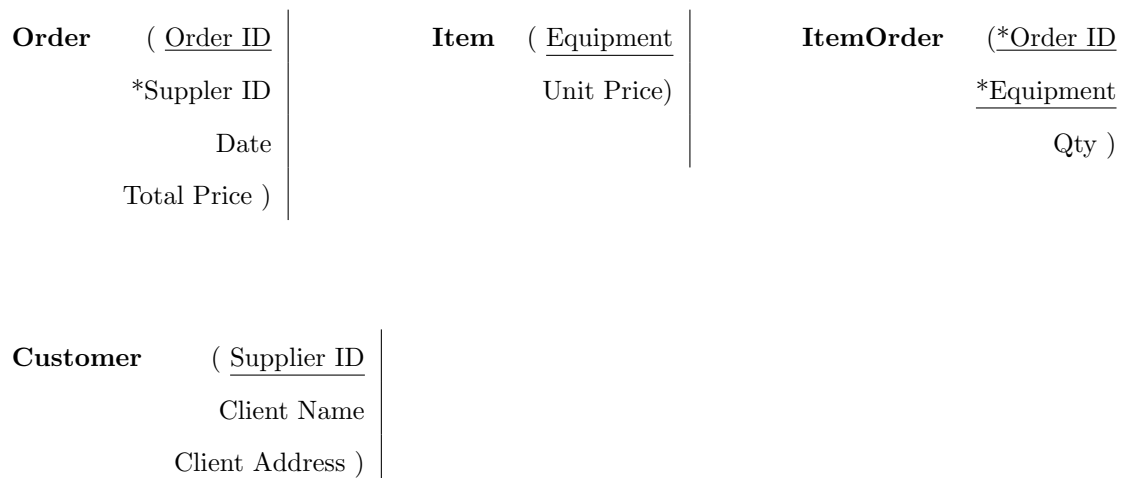
Primary Key	Attribute	Transitive Dependency?
Order ID	Supplier ID	Yes: Supplier ID can be found if we know Client Name or Address
Order ID	Client Name	Yes: Client Name can be found if we know Supplier ID or Address
Order ID	Client Address	Yes: Client Address can be found if we know Client Name or Supplier ID
Order ID	Date	No: Only dependent on Primary Key
Order ID	Total Cost	No: Only dependent on Primary Key

Using this table I have identified that *Supplier ID*, *Client Name*, and *Client Address* all have transitive dependencies, and need to be moved to a new entity.

Step 2: Entity, Relationship and Key

I created a new entity called *Customer*, and moved the three attributes with transitive dependencies into it. I then made *Supplier ID* the Primary Key and created a relationship between the *Customer* and *Order* entities by repeating the *Supplier ID* attribute as a foreign key in the *Order* entity.

Entities in Third Normal Form: Diagram



Entities in Third Normal Form: With Data

Order entity

Order ID	Supplier ID	Date	Total Price
CON-1234	032	16/04/2015	23.00
CON-2237	168	14/12/2014	99.00
CON-2356	169	12/02/2015	280.00
CON-3664	527	16/01/2015	36.00

Item entity

Equipment	Unit Price
3/4" EMT	50.00
3/4" Locknut	1.50
4x8x3/4 Cos Plywood	10.00
Butterfly Valve	5.00
Duplex Ivy Rec	100.00
Sch 40 Blk Pipe	20.00
Thin Stranded Copper Wire	6.00

ItemOrder entity

Order ID	Equipment	Qty
CON-1234	3/4" Locknut	2
CON-1234	Sch 40 Blk Pipe	1
CON-2237	3/4" Locknut	6
CON-2237	Butterfly Valve	2
CON-2237	Sch 40 Blk Pipe	4
CON-2356	3/4" EMT	2
CON-2356	4x8x3/4 Cos Plywood	2
CON-2356	Duplex Ivy Rec	1
CON-2356	Sch 40 Blk Pipe	3
CON-3664	Thin Stranded Copper Wire	6

Customer entity

Supplier ID	Client Name	Client Address
032	Grand Design Ltd	Unit 12, Stoneleigh Park, CV8 2UV
168	Coventry Building Services Ltd	Units 2-4, Binley Industrial Estate, CV3 2WL
169	Ricoh Builds Ltd	34, Lythalls La Industrial Estate, CV6 6RG
527	Allied Construction Ltd	32-34, Bilton Industrial Estate, CV3 5YB

Activity 4: ER Diagram

Task 2: Database Development

For this task I will provide the SQL statement I used to complete each question activity, I will then explain each part of the SQL statement, and give screen-shots of before and after the statement was executed (if applicable).

Each SQL statement was written and tested using ORACLE 11g Express Edition and ORACLE Application Express.

Question 1

To begin I created the two tables which have no dependency on any other table: *Aircraft* and *Airline*

Aircraft

Listing 1: CREATE AIRCRAFT

```
1 CREATE TABLE Aircraft
2 (   aircraft_code VARCHAR2(5) PRIMARY KEY,
3     aircraft_type VARCHAR2(30) NOT NULL,
4     aircraft_price NUMBER(11,2) NOT NULL,
5 );
```

The statement begins with '*CREATE TABLE Aircraft*' which will create a table called *aircraft*. Inside the brackets the three fields, their data types, and any constraints are listed. Below is a table which will explain why I chose these data types and constraints for each field.

Identifier	Data-Type	Constraint	Explanation
aircraft_code	VARCHAR2(5)	PRIMARY KEY	All aircraft_codes start with 'C' and are followed by up to four numerical digits. This field will be used as the Primary Key of the table.
aircraft_type	VARCHAR(30)	NOT NULL	Contains a combination of alphanumeric characters of up to 30 characters in length. This field cannot be left empty.
aircraft_price	NUMBER(11,2)	NOT NULL	Stores large numeric values, with a precision of 11 and a scale of 2. This allows for prices up to 99 Thousand Million and two decimal places. This field cannot be left empty.

Airline

Listing 2: CREATE AIRLINE

```
1 CREATE TABLE Airline
2 (   airline_code CHAR(4) PRIMARY KEY,
3     airline_name VARCHAR2(20) NOT NULL,
4     airline_address VARCHAR2(60) NOT NULL,
5     airline_city VARCHAR2(15) NOT NULL,
6     airline_country VARCHAR2(15) NOT NULL
7 );
```

Identifier	Data-Type	Constraint	Explanation
airline_code	CHAR(4)	PRIMARY KEY	A combination of 4 alphanumeric characters. airline_code is also the Primary Key
airline_name	VARCHAR2(20)	NOT NULL	This field allows a variable length of characters, up to a maximum of 20. This field cannot be left empty.
airline_address	VARCHAR2(60)	NOT NULL	This field allows for a combination of alphanumeric characters up to a maximum length of 60. This field cannot be left empty.
airline_city	VARCHAR2(15)	NOT NULL	This field allows a combination of alphanumeric characters up to a maximum length of 15. This field cannot be left empty.
airline_country	VARCHAR2(15)	NOT NULL	This field allows for a combination of alphanumeric characters up to a maximum length of 15. This field cannot be left empty.

Purchase_Order

Listing 3: CREATE PURCHASE_ORDER

```
1 CREATE TABLE purchase_order
2 ( purchase_order_no NUMBER(3,0) PRIMARY KEY ,
3   airline_code CHAR(4) NOT NULL,
4   date_of_purchase DATE NOT NULL,
5   CONSTRAINT FK_1 FOREIGN KEY (airline_code) REFERENCES airline(airline_code)
6 );
```

Identifier	Data-Type	Constraint	Explanation
purchase_order_no	NUMBER(3,0)	PRIMARY KEY	This field allows for numeric input with a precision of 3 and a scale of 0, this allows values in the range of 001 to 999. This field also acts as the Primary Key.
airline_code	CHAR(4)	NOT NULL, FOREIGN KEY	A combination of 4 alphanumeric characters. This field is a foreign key; Referencing the <i>airline_code</i> field from the <i>Airline</i> table.
date_of_purchase	DATE	NOT NULL	

Ordered_Aircraft

Listing 4: CREATE ORDERED_AIRCRAFT

```

1 CREATE TABLE ordered_aircraft
2 (
3 purchase_order_no NUMBER(3,0) NOT NULL,
4 aircraft_code VARCHAR2(5) NOT NULL,
5 aircraft_quantity NUMBER(3,0) NOT NULL,
6 CONSTRAINT PK_ORDERCODE PRIMARY KEY(purchase_order_no, aircraft_code),
7 CONSTRAINT FK_PO FOREIGN KEY (purchase_order_no)
8 REFERENCES purchase_order(purchase_order_no);
9 CONSTRAINT FK_AC FOREIGN KEY (aircraft_code)
10 REFERENCES aircraft(aircraft_code)
11 );

```

Identifier	Data-Type	Constraint	Explanation
purchase_order_no	NUMBER(3,0)	NOT NULL, PRIMARY KEY, FOREIGN KEY	This field allows for numeric input with a precision of 3 and a scale of 0. It is a foreign key referencing; <i>purchase_order_no</i> from <i>purchase_order</i>
aircraft_code	VARCHAR2(5)	NOT NULL, PRIMARY KEY, FOREIGN KEY	This field allows for alphanumeric input up to a maximum length of 5. It is part of a composite key and is also a foreign key referencing <i>aircraft_code</i> from <i>aircraft</i> .
aircraft_quantity.	NUMBER (3,0)	NOT NULL	This field allows for 3 digit numbers. This field cannot be left blank.

Question 2

Aircraft

Listing 5: INSERT INTO *AIRCRAFT*

```

1 INSERT ALL
2 INTO Aircraft VALUES ('C800' , 'CU-800 Commuter' , 8000000000)
3 INTO Aircraft VALUES ('C8000' , 'CU-8000 Stratocruiser' , 2290000000)
4 INTO Aircraft VALUES ('C9000' , 'CU-9000 Mesocruiser' , 3000000000)
5 INTO Aircraft VALUES ('C24' , 'CU-24 Slipstream' , 2100000000)
6 INTO Aircraft VALUES ('C8' , 'CU-8X Heavy Lifter' , 1000000000)
7 INTO Aircraft VALUES ('C10' , 'CU-10 Exolinear' , 4000000000)
8 INTO Aircraft VALUES ('C900' , 'CU-900 Commuter' , 1900000000)
9 INTO Aircraft VALUES ('C80' , 'CU-80 Cloud Hopper' , 860000000)
10 INTO Aircraft VALUES ('C6' , 'CU-6X Rapid Lifter' , 2800000000)
11 INTO Aircraft VALUES ('C22' , 'CU-22 Executive Jet' , 1100000000)
12 INTO Aircraft VALUES ('C5' , 'CU-X5 Spaceplane' , 6000000000)
13 SELECT 1 FROM DUAL;
```

The SQL statement uses the following pattern:

INSERT ALL...INTO (table identifier) VALUES (data) .

This allows me to input multiple records into the table simultaneously. '*INTO Aircraft*' specifies that I am going to insert the following values into the *Aircraft* table.

The text in brackets following the word *VALUES* is the data which is input into that record.

'*SELECT 1 FROM DUAL*' is a requirement of *INSERT ALL*, so that it can function correctly.


EDIT	AIRCRAFT_CODE	AIRCRAFT_TYPE	AIRCRAFT_PRICE
	C800	CU-800 Commuter	800000000
	C8000	CU-8000 Stratocruiser	2290000000
	C9000	CU-9000 Mesocruiser	3000000000
	C24	CU-24 Slipstream	2100000000
	C8	CU-8X Heavy Lifter	1000000000
	C10	CU-10 Exolinear	4000000000
	C900	CU-900 Commuter	1900000000
	C80	CU-80 Cloud Hopper	860000000
	C6	CU-6X Rapid Lifter	2800000000
	C22	CU-22 Executive Jet	1100000000
	C5	CU-X5 Spaceplane	6000000000
row(s) 1 - 11 of 11			

AirlineListing 6: INSERT INTO *AIRLINE*

```

1 INSERT ALL
2
3 INTO Airline VALUES ('BA07','British Airways PLC',
4 'Waterside, PO Box 365, Harmondsworth, UB7 0GB' , 'London', 'England')
5
6 INTO Airline VALUES ('UA09','United Airlines',
7 '77 W. Wacker Drive, Chigaco, IL 60601, United States', 'Chicago', 'United States')
8
9 INTO Airline VALUES ('AI06','Air India' ,
10 'Air-India Building, Nairman Point, Mumbai, 400 021' , 'Mumbai', 'India')
11
12 INTO Airline VALUES ('AC05','Air Coventry LTD' ,
13 'Coventry Univeristy, Priory Street, COventry, CV1 5FB' , 'Coventry', 'England')
14
15 INTO Airline VALUES ('RM04','Royal Mail Group',
16 '100 Victoria Embankment, London, EC4y 0HQ' , 'London', 'England')
17
18 INTO Airline VALUES ('IR01','Iran Air',
19 '1 Valiasr Street, Tehran' , 'Tehran', 'Iran')
20
21 SELECT 1 FROM DUAL;







```

EDIT	AIRLINE_CODE	AIRLINE_NAME	AIRLINE_ADDRESS	AIRLINE_CITY	AIRLINE_COUNTRY
	BA07	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	London	England
	UA09	United Airlines	77 W. Wacker Drive, Chigaco, IL 60601, United States	Chicago	United States
	AI06	Air India	Air-India Building, Nairman Point, Mumbai, 400 021	Mumbai	India
	AC05	Air Coventry LTD	Coventry Univeristy, Priory Street, COventry, CV1 5FB	Coventry	England
	RM04	Royal Mail Group	100 Victoria Embankment, London, EC4y 0HQ	London	England
	IR01	Iran Air	1 Valiasr Street, Tehran	Tehran	Iran
row(s) 1 - 6 of 6					

Purchase_OrderListing 7: INSERT INTO *PURCHASE_ORDER*

```

















1 INSERT ALL
2
3 INTO Purchase_Order VALUES (689,'BA07',TO_DATE ('28/04/2012', 'dd/mm/yyyy'))
4 INTO Purchase_Order VALUES (789,'UA09',TO_DATE ('24/06/2011', 'dd/mm/yyyy'))
5 INTO Purchase_Order VALUES (800,'AI06',TO_DATE ('30/07/2013', 'dd/mm/yyyy'))
6 INTO Purchase_Order VALUES (898,'AC05',TO_DATE ('06/06/2012', 'dd/mm/yyyy'))
7 INTO Purchase_Order VALUES (900,'RM04',TO_DATE ('03/11/2012', 'dd/mm/yyyy'))
8 INTO Purchase_Order VALUES (901,'IR01',TO_DATE ('03/11/2014', 'dd/mm/yyyy'))
9
10 SELECT 1 FROM DUAL;
```


EDIT	PURCHASE_ORDER_NO	AIRLINE_CODE	DATE_OF_PURCHASE
	689	BA07	04/28/2012
	789	UA09	06/24/2011
	800	AI06	07/30/2013
	898	AC05	06/06/2012
	900	RM04	11/03/2012
	901	IR01	11/03/2014
row(s) 1 - 6 of 6			

Ordered_AircraftListing 8: INSERT INTO *ORDERED_AIRCRAFT*

```

1 INSERT ALL
2 INTO Ordered_Aircraft VALUES( 689, 'C800',14)
3 INTO Ordered_Aircraft VALUES( 689, 'C8000',8)
4 INTO Ordered_Aircraft VALUES( 689, 'C9000',5)
5 INTO Ordered_Aircraft VALUES( 689, 'C24',14)
6 INTO Ordered_Aircraft VALUES( 689, 'C8',7)
7 INTO Ordered_Aircraft VALUES( 789, 'C10',4)
8 INTO Ordered_Aircraft VALUES( 789, 'C900',30)
9 INTO Ordered_Aircraft VALUES( 789, 'C8000',8)
10 INTO Ordered_Aircraft VALUES( 800, 'C80',9)
11 INTO Ordered_Aircraft VALUES( 800, 'C6',11)
12 INTO Ordered_Aircraft VALUES( 800, 'C22',32)
13 INTO Ordered_Aircraft VALUES( 898, 'C5',6)
14 INTO Ordered_Aircraft VALUES( 898, 'C24',12)
15 INTO Ordered_Aircraft VALUES( 898, 'C800',25)
16 INTO Ordered_Aircraft VALUES( 898, 'C10',6)
17 INTO Ordered_Aircraft VALUES( 900, 'C8',8)
18 INTO Ordered_Aircraft VALUES( 900, 'C6',25)
19 INTO Ordered_Aircraft VALUES( 900, 'C80',8)
20 INTO Ordered_Aircraft VALUES( 901, 'C80',10)
21
22
23 SELECT 1 FROM DUAL;
```

EDIT	PURCHASE_ORDER_NO	AIRCRAFT_CODE	AIRCRAFT_QUANTITY
	689	C800	14
	689	C8000	8
	689	C9000	5
	689	C24	14
	689	C8	7
	789	C10	4
	789	C900	30
	789	C8000	8
	800	C80	9
	800	C6	11
	800	C22	32
	898	C5	6
	898	C24	12
	898	C800	25
	898	C10	6
row(s) 1 - 15 of 19 			

EDIT	PURCHASE_ORDER_NO	AIRCRAFT_CODE	AIRCRAFT_QUANTITY
	900	C8	8
	900	C6	25
	900	C80	8
	901	C80	10
			 row(s) 16 - 19 of 19

Question 3

This question requires me to find the; Average, Minimum and Maximum price for all aircraft bought after '01/01/2012'.

Listing 9: AVG() MIN() MAX()

```

1 SELECT AVG(ordered_aircraft.aircraft_quantity * aircraft_price) AS Average_Price,
2 MIN(ordered_aircraft.aircraft_quantity * aircraft_price) AS Minimum_Price,
3 MAX(ordered_aircraft.aircraft_quantity * aircraft_price) AS Maxium_Price
4 FROM ordered_aircraft JOIN aircraft
5 ON ordered_aircraft.aircraft_code = aircraft.aircraft_code
6 JOIN purchase_order
7 ON ordered_aircraft.purchase_order_no = purchase_order.purchase_order_no
8 WHERE purchase_order.date_of_purchase > '01/01/2012';

```

Firstly, to work out the price for all the aircraft bought, I multiply the quantity of aircraft ordered (*aircraft_quantity*) by the price of that particular aircraft (*aircraft_price*). To find the average, minimum and maximum of these prices then I need to use this calculation in conjunction with the *AVG()*, *MIN()* and *MAX()* SQL functions respectively. These functions will perform this calculation on every record in the specified tables and return the correct value. Once I selected the values, I used the *AS* keyword to label the output columns.

This can be seen in line 1 - 3 in the code above.

In total, this SQL statement uses three tables *ordered_aircraft* and *aircraft* for the calculations, and *purchase_order* for the conditional statement.

To be able to use the data from multiple tables, I need to use *JOINS*. *JOINS* allow me to combine common records from two or more tables by using the syntax structure;

JOIN table name *ON* table1.attribute = table2.attribute

For this problem I used two joins;

1. *JOIN* aircraft *ON* ordered_aircraft.aircraft_code = aircraft.aircraft_code
2. *JOIN* purchase_order *ON* ordered_aircraft.purchase_order_no = purchase_order.purchase_order_no

Join number One is used to select the correct relation attributes from both required tables.

Join number Two is used to ensure that the *WHERE* condition relates to the selected attributes.

This allowed me to select all the values I needed and also make sure that the value used in each calculation were from the same record.

Below is a screenshot of the statements' output.

AVERAGE_PRICE	MINIMUM_PRICE	MAXIUM_PRICE
4233375000	688000000	36000000000

Question 4

This question required me to: "Display the purchase order number, date, airline name, address, airline country for airlines where the total cost of an order is less than 10,000 million pounds. Your results should be in descending alphabetical order based on the airline code".

Listing 10: Question 4

```

1 SELECT
2 purchase_order.purchase_order_no,
3 purchase_order.date_of_purchase,
4 airline.airline_name,
5 airline.airline_address,
6 airline.airline_country,
7 ordered_aircraft.aircraft_quantity * aircraft.aircraft_price AS total_cost
8 FROM airline
9 JOIN purchase_order
10 ON airline.airline_code=purchase_order.airline_code
11 JOIN ordered_aircraft
12 ON purchase_order.purchase_order_no=ordered_aircraft.purchase_order_no
13 JOIN aircraft
14 ON ordered_aircraft.aircraft_code=aircraft.aircraft_code
15 WHERE ordered_aircraft.aircraft_quantity * aircraft.aircraft_price < 10000000000
16 ORDER BY airline.airline_code DESC;
```

Lines 1 - 8 above I am using a **SELECT ... FROM** statement to select the required attributes from the tables. However not all of these attributes are from the same table, so I used multiple **JOIN** statements (lines 9 - 14) to combine the common attributes. On line 15 I used a **WHERE** keyword to specify the required condition (Total cost < 10,000 Million pounds). Finally on Line 16, I sorted the results in descending order on *airline_code* by using **ORDER BY ...DESC**.

Below is a screenshot of the statements' output.

PURCHASE_ORDER_NO	DATE_OF_PURCHASE	AIRLINE_NAME	AIRLINE_ADDRESS	AIRLINE_COUNTRY	TOTAL_COST
789	06/24/2011	United Airlines	77 W. Wacker Drive, Chicago, IL 60601, United States	United States	1600000000
789	06/24/2011	United Airlines	77 W. Wacker Drive, Chicago, IL 60601, United States	United States	5700000000
789	06/24/2011	United Airlines	77 W. Wacker Drive, Chicago, IL 60601, United States	United States	1832000000
900	11/03/2012	Royal Mail Group	100 Victoria Embankment, London, EC4y 0HQ	England	7000000000
900	11/03/2012	Royal Mail Group	100 Victoria Embankment, London, EC4y 0HQ	England	8000000000
900	11/03/2012	Royal Mail Group	100 Victoria Embankment, London, EC4y 0HQ	England	6880000000
901	11/03/2014	Iran Air	1 Valiasr Street, Tehran	Iran	8600000000
689	04/28/2012	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	England	7000000000
689	04/28/2012	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	England	2940000000
689	04/28/2012	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	England	1500000000
689	04/28/2012	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	England	1832000000
689	04/28/2012	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	England	1120000000
800	07/30/2013	Air India	Air-India Building, Nairman Point, Mumbai, 400 021	India	3520000000
800	07/30/2013	Air India	Air-India Building, Nairman Point, Mumbai, 400 021	India	7740000000
800	07/30/2013	Air India	Air-India Building, Nairman Point, Mumbai, 400 021	India	3080000000
898	06/06/2012	Air Coventry LTD	Coventry University, Priory Street, COventry, CV1 5FB	England	2520000000
898	06/06/2012	Air Coventry LTD	Coventry University, Priory Street, COventry, CV1 5FB	England	2000000000
898	06/06/2012	Air Coventry LTD	Coventry University, Priory Street, COventry, CV1 5FB	England	2400000000

Question 5

This question required me to: "Display how many aircraft were ordered by British Airways in total and by each different airplane type."

Listing 11: Question5

```
1 SELECT ordered_aircraft.aircraft_quantity AS Total_aircraft_ordered,  
2 aircraft.aircraft_type  
3 FROM ordered_aircraft  
4 JOIN aircraft  
5 ON ordered_aircraft.aircraft_code=aircraft.aircraft_code  
6 JOIN purchase_order  
7 ON ordered_aircraft.purchase_order_no=purchase_order.purchase_order_no  
8 JOIN airline  
9 ON purchase_order.airline_code=airline.airline_code  
10 WHERE airline.airline_code ='BA07';
```

Lines 1 - 3 use **SELECT ... FROM** to select the two rows needed for this task.

Lines 4 - 9 contains the joins required to combine the related data in the tables.

Lines 10 uses **WHERE** to define a condition. In this particular case it only selects data where airline_code is equal to BA07 , which is the airline code for "British airways PLC".

Below is a screenshot of the statements' output.

TOTAL_AIRCRAFT_ORDERED	AIRCRAFT_TYPE
14	CU-24 Slipstream
7	CU-8X Heavy Lifter
14	CU-800 Commuter
8	CU-8000 Stratocruiser
5	CU-9000 Mesocruiser

Question 6

This question required me to: "Produce a list of all the orders from all the airlines. The list should show the airline code, followed by the order number, followed by the code of the aircraft ordered, followed by the quantity of aircraft ordered and then the total cost of that order. The list should be arranged by airline code in descending order."

Listing 12: Question 6

```

1 SELECT
2   airline.airline_code,
3   ordered_aircraft.purchase_order_no,
4   aircraft.aircraft_code,
5   ordered_aircraft.aircraft_quantity,
6   (ordered_aircraft.aircraft_quantity * aircraft.aircraft_price) AS total_price
7 FROM airline
8 JOIN purchase_order
9   ON purchase_order.airline_code=airline.airline_code
10 JOIN ordered_aircraft
11   ON purchase_order.purchase_order_no=ordered_aircraft.purchase_order_no
12 JOIN aircraft
13   ON ordered_aircraft.aircraft_code=aircraft.aircraft_code
14 ORDER BY airline.airline_code DESC;
```

Lines 1 - 7 use **SELECT ... FROM** to select the five rows needed for this task and specify their location.

Lines 8 - 13 join the tables so to combined the related data to achieve contiguous output.

Line 14 **ORDER BY ...DESC** to sort the data in descending order via the *airline_code* column.

Below is a screenshot of the statements' output.

AIRLINE_CODE	PURCHASE_ORDER_NO	AIRCRAFT_CODE	AIRCRAFT_QUANTITY	TOTAL_PRICE
UA09	789	C900	30	5700000000
UA09	789	C10	4	1600000000
UA09	789	C8000	8	1832000000
RM04	900	C6	25	7000000000
RM04	900	C80	8	6880000000
RM04	900	C8	8	8000000000
IR01	901	C80	10	8600000000
BA07	689	C24	14	2940000000
BA07	689	C800	14	11200000000
BA07	689	C8	7	7000000000
BA07	689	C8000	8	18320000000
BA07	689	C9000	5	15000000000
AI06	800	C22	32	35200000000
AI06	800	C6	11	30800000000
AI06	800	C80	9	7740000000
AC05	898	C800	25	20000000000
AC05	898	C5	6	360000000000
AC05	898	C24	12	25200000000
AC05	898	C10	6	24000000000

Question 7

This question required me to : ". Display the order details, airline details and the aircraft details where more than 10 aircraft of a specific type were ordered."

Listing 13: Question 7

```

1  SELECT purchase_order.*,
2  ordered_aircraft.aircraft_quantity,
3  airline.airline_name,
4  airline.airline_address,
5  airline.airline_city,
6  airline.airline_country,
7  ordered_aircraft.aircraft_code,
8  aircraft.aircraft_type,
9  aircraft.aircraft_price
10 FROM purchase_order
11 JOIN airline
12 ON airline.airline_code=purchase_order.airline_code
13 JOIN ordered_aircraft
14 ON purchase_order.purchase_order_no=ordered_aircraft.purchase_order_no
15 JOIN aircraft
16 ON ordered_aircraft.aircraft_code=aircraft.aircraft_code
17 WHERE ordered_aircraft.aircraft_quantity > 10;

```

Lines 1 - 10 select the required rows from the data tables.

Lines 11 - 16 Joins the tables on the common data elements.

Line 17 is a condition statement, where the *aircraft_quantity* is greater than ten.

Below is a screenshot of the statements' output.

PURCHASE_ORDER_NO	AIRLINE_CODE	DATE_OF_PURCHASE	AIRCRAFT_QUANTITY	AIRLINE_NAME	AIRLINE_ADDRESS	AIRLINE_CITY	AIRLINE_COUNTRY	AIRCRAFT_CODE	AIRCRAFT_TYPE	AIRCRAFT_PRICE
898	AC05	06/06/2012	25	Air Coventry LTD	Coventry University, Priory Street, Coventry, CV1 5FB	Coventry	England	C800	CU-800 Commuter	80000000
689	BA07	04/28/2012	14	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	London	England	C800	CU-800 Commuter	80000000
898	AC05	06/06/2012	12	Air Coventry LTD	Coventry University, Priory Street, Coventry, CV1 5FB	Coventry	England	C24	CU-24 Slipstream	21000000
689	BA07	04/28/2012	14	British Airways PLC	Waterside, PO Box 365, Harmondsworth, UB7 0GB	London	England	C24	CU-24 Slipstream	21000000
789	UA09	06/24/2011	30	United Airlines	77 W. Wacker Drive, Chicago, IL 60601, United States	Chicago	United States	C900	CU-900 Commuter	19000000
900	RM04	11/03/2012	25	Royal Mail Group	100 Victoria Embankment, London, EC4y 0HQ	London	England	C6	CU-6X Rapid Lifter	28000000
800	AI06	07/30/2013	11	Air India	Air-India Building, Nairman Point, Mumbai, 400 021	Mumbai	India	C6	CU-6X Rapid Lifter	28000000
800	AI06	07/30/2013	32	Air India	Air-India Building, Nairman Point, Mumbai, 400 021	Mumbai	India	C22	CU-22 Executive Jet	11000000


Question 8

This question required me to: "We have discovered an error in our spreadsheet. The C800 aircraft should have been 100 million not 80 million please update the database based on this change."

Listing 14: Question 8

```
1 UPDATE aircraft
2 SET aircraft_price=100000000
3 WHERE aircraft_code='C800'
```

Below is a screen shot of the updated record.

EDIT	AIRCRAFT_CODE	AIRCRAFT_TYPE	AIRCRAFT_PRICE
	C800	CU-800 Commuter	100000000


Question 9

This question required me to: ". Air Coventry LTD has changed its name to Coventry University Airways please update the information."

Listing 15: Question 9

```
1 UPDATE airline
2 SET airline_name='Coventry University Airways'
3 WHERE airline_name='Air Coventry LTD'
```

Below is a screen shot of the updated record.

	AC05	Coventry University Airways	Coventry Univeristy, Priory Street, COventry, CV1 5FB	Coventry	England
---	------	-----------------------------	---	----------	---------

Task 3 : Poster of Ethics

Task 4 : Amazon recommendation system

What is a Recommendation System?

Recommendation systems are commonly described as:

"A way of presenting information on items and products that are likely to be of interest to the reader."
[1]

Recommendation systems work via the use of Data Mining. Data Mining is a process where data is gathered and analysed to find specific patterns. These patterns can provide meaningful and insightful data which can then be relayed to the consumer.

The data gathered for a recommendation system comes in many varieties and is often tailored to the recommendation systems' main purpose. For Example, a system may collect data on a users' Hobbies, Interests, Opinions, Habits, Recent search history, Recent Purchase History and many more areas.

MOAR OF THIS, EXPANDED EXAMPLES E-COMMERCE, BOOK RECOMMENDATIONS, ETC

Data Gathering and Preparation

Amazon is an incredibly popular E-Commerce website which caters to millions of customers everyday. This is partially due to its intricate and extensive data mining operations which result in a very effective recommendation system. The effectiveness of this system can be seen by looking at their percentage sales increases;

"The company reported a 29% sales increase to \$12.83 billion during its second fiscal quarter, up from \$9.9 billion during the same time last year." [2]

Data

To make the recommendation system as effective as possible, Amazon must collect a huge amount of data on each customer and more importantly they must collect the right data. In this section I will explore what kind of data this recommendation system should collect.

At Amazon, the recommendation system is used to personalize the shopping environment for each customer, so the store changes drastically depending on the customer.

Completed Purchases

Products the customer has spent money on.

Incomplete purchases

Abandoned shopping carts or specific items dropped from the cart.

Personal item ratings/reviews

Ratings will change the recommendations depending on the quality of the rating.

Demographic information

Such as Location, Age, Profession, Type of education etc... will offer different recommendations depending on which demographic criteria the customer meets.

Number of views

How many times was thaty particular item, or similar items viewed before purchase.

Time spent before purchase

How long it takes for a customer to purchase an item

Click-Through Rate

How often a customer clicks on a recommendation.

Conversion rate

How often the customer purchases an item that has been recommended.

Referrals

Did the customer arrive here from a different website, if so, that website could infer interest in specific items.

All of these possible information sources can be categorized into three main categories:

Demographic Data

This is data about the customer, which can be used to identify them. e.g Age, Gender, Social class and Location.

Item Data

This is data that can identify specific items or groups of related items. This may include keywords, tags, Genres and types.

User-Item Data

This is explicit data, which is provided by the customer in the form of ratings and reviews. Ratings and reviews act as a linear scale on which a recommender system can use to recommend other items.

All of these types of data are in a web-format, meaning that all of the data here is collected via the consumers actions on the web page. The types of data that can be gathered to include other external sources and formats. For example the recommendation system could be expanded to scrape information from the customers various social media profiles to gather more information.

Benefits and limitations of a Recommendation system

Good recommendation systems drastically improve e-commerce sales, but there are also some downsides. In this section I will explore both the benefits and limitations in detail.

Limitations

Complexity

Making a system that produces relevant recommendations requires a lot of work. Primarily because there are so many variables that go into producing the most basic of recommendations.

It takes incredibly complex algorithms and a large amount of processing power to produce any recommendations at all. If the recommendation algorithm isn't as complex as it needs to be, or it doesn't use the right comparisons then it will only produce recommendations that are either very obvious or not of any interest at all.

Amount of Data For recommendation systems to operate at their most effective, they need a large amount of accurate data. Not only do they need customer data, but also item data. Before a recommendation system is implemented a large amount of user data needs to be collected. This can prove difficult because sometimes that user data just isn't available. The more data a company can collect about their customers and about their own recommendation system results, the more accurate it will be.

Data Accuracy Data, by its nature is a rapidly expanding resource. Whilst the collection of data is a relatively simple process, deciding whether or not the data is relevant is far more complex. Hopefully, depending on the way the data is gathered, all of it will be relevant, but this is not always the case and valuable time and resources are used collecting and analysing this useless data.

Not only may some of the data be irrelevant, some of it may be contradictory. For example if a customer changes some of his/her preferences or changes demographic, also shopping trends are always in flux i.e what was popular last week might not be the case a month from now. This inaccurate data will make the recommendation system less effective.

Benefits

Sales

An accurate recommendation system has the potential to increase sales and profits. This is done by giving the customer recommendations that they are interested in but did not plan to purchase, this increases average order value.

Once a customer has received personalized recommendations, overall customer retention (customer loyalty) increases.

Trends and Market data

The recommendation system doesn't just need a lot of data, it can also produce valuable data about shopping habits and market trends. This data can help drive the direction of the business to improve future sales and sales forecasting.

Real-Time Data

Similar to the section above, having data coming into the system in real time has many advantages. The data being collected is coming into the system in real time, this means that very little 'Guess-Work' needs to be done with predictions. The influx of new data means that recommendations can be updated almost instantaneously as soon as that data is available. Using current data minimises the potential for inaccurate or irrelevant predictions.

Automation

The recommendation system holds data about every customer, it is continuously analysing their actions. Because of this, the users are effectively organizing their own content of the web page and deciding what they see in the future. The customer may not be away of this at all times, but this is very beneficial for the company. With the only participants being the system and the customers, a vast amount of the organisation and customer maintenance on the business side is reduced .

Analysis

In this section I will explore how the types of data gathered above can be analysed to produce meaningful recommendations via *Association analysis*.

Association analysis defines a broad spectrum of analytical techniques that focus on using complex algorithms to analyse large data sets to produce statistical rules that can explain the patterns within the data.

Collaborative filtering

Collaborative filtering is done by collecting specific data about the items that a customer has bought, and then comparing this data to that of other users. The recommendations produced are items that are liked by a user with similar purchase/rating data.

The data needed for this approach is the explicit *User-Item* data, which consist of ratings and reviews. Each time a customer purchases or positively rates a product, it is added to their 'profile' vice versa for negatively rated items. The customer 'profiles' are what are compared to each other and the similarities and differences of the 'profiles' are used to determine different products.

For example if *User A* positively rates *product 1* and *product 2*.

And if *User B* also positively rates *Product 1* and *2* and also positively rates *Product 3*, then the system will determine that *User A* would also like *Product 3* and recommend it to them.

This is a Model-Based approach, where customers are grouped (clustered) by similar rating and preferences. From these groups, a large dataset is made which the system analyses to produce a probabilistic model which can be used to determine an expected rating for products.

The benefits to this approach are that no item data is needed to produce results and it also has a high probability to recommend 'outside the box' products which the customer had not previous knowledge of.

The Model-based approach has the potential to be very fast and 'resource friendly'.

There are however, some disadvantages, such as how the size of the data gathered affects the recommendations. Often customers do not provide ratings or review for purchased items and new customers do not have any at all, which limits the systems effectiveness at identifying customers with similar 'profiles'. When this scenario occurs, the approach loses it speed and resource advantage.

The Demographic approach

Bibliography

References for Task 3: A Poster of Ethics

References for Task 4: A Recommendation system

1. http://www.webopedia.com/TERM/R/recommender_systems.
2. <http://fortune.com/2012/07/30/amazons-recommendation-secret/>
3. <http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf>
4. <http://stackoverflow.com/questions/2323768/how-does-the-amazon-recommendation-feature>
5. https://www.few.vu.nl/nl/Images/werkstuk-hiralall_tcm38-202691.pdf
6. http://webcache.googleusercontent.com/search?q=cache:X2_XVw9qTicJ:readwrite.com/2009/01/28/5_problems_of_recommender_systems+&cd=1&hl=en&ct=clnk&gl=uk
7. https://www.few.vu.nl/nl/Images/werkstuk-hiralall_tcm38-202691.pdf
8. http://www.uie.com/articles/recommendation_systems/
9. <http://infolab.stanford.edu/~ullman/mmds/ch9.pdf>