NCC Level-5DC Diploma in Computing

Database Design and Development

Candidates Name : Fatema Akter

ID No : 00154713

Module Title : Database Design and Development

Assignment Title : Forest Film

Examination Cycle: June 2016

Candidate attempting to gain an unfair advantage or colluding in anyway whatsoever (other than on joint assignments) are liable to be disqualified. Plagiarism is an offence.

Expected candidate time allocation: 35 to 40 hours

Mark	Moderated Mark	Final Mark

Marker's comment:

Moderator's comment:

Statement of Confirmation of Own Work

Programmed /qualification name: Database Design and Development

Student Declaration:

I have read and understood the NCC Education's policy on Academic Dishonesty and Plagiarism.

I can confirm the following details**:**

Student ID/Registration number : 00154713

Name : Fatema Akter

Center Name : Daffodil International Academy.

Module Name : Database Design and Development

Assignment Title : Forest Film

Number of Words : 1,532

I confirm that this is my own and that I have not plagiarized any part of it. I have also noted the assessment criteria and pass mark for assignments.

Due Date: 12/04/2016

Submitted Date: 11/04/2016

Student Signature: Fatema Akter

ACKNOWLEDGEMENT

At the beginning I would like to render thanks to the almighty Allah. And so I would wish to show my special thanks, gratitude to my teacher ***Mr. Shomon Hossian*** well as all other teachers. I did a great deal of research and I came to know about so many recalls and it helped to increase my knowledge.

Once more, I would wish to give thanks all of them who helped me to complete this Assignment.

Table of Contents

Introduction:

Now I am going to speak assignment ***Database Design and development,*** in this assignment topic the ***Forest File.*** The assignment ***contains four parts***, such as ***Task-1: Design***, ***Task-2: Data and queries Task-3: Derived data and Task-4: Evaluation.*** This assignment wills opportunity me to show my ***knowledge*** and understanding of ***Database Design and development***.

Task-1

Design:

(a) Entity Relationship Model:

Record for the entity name:

Here I have used some entity in this database model including this:

- Client_tab
- Location_tab
- Production_property_tab
- Production_staff_tab
- Production_tab
- Product_tab
- Properties_tab
- Staff_tab

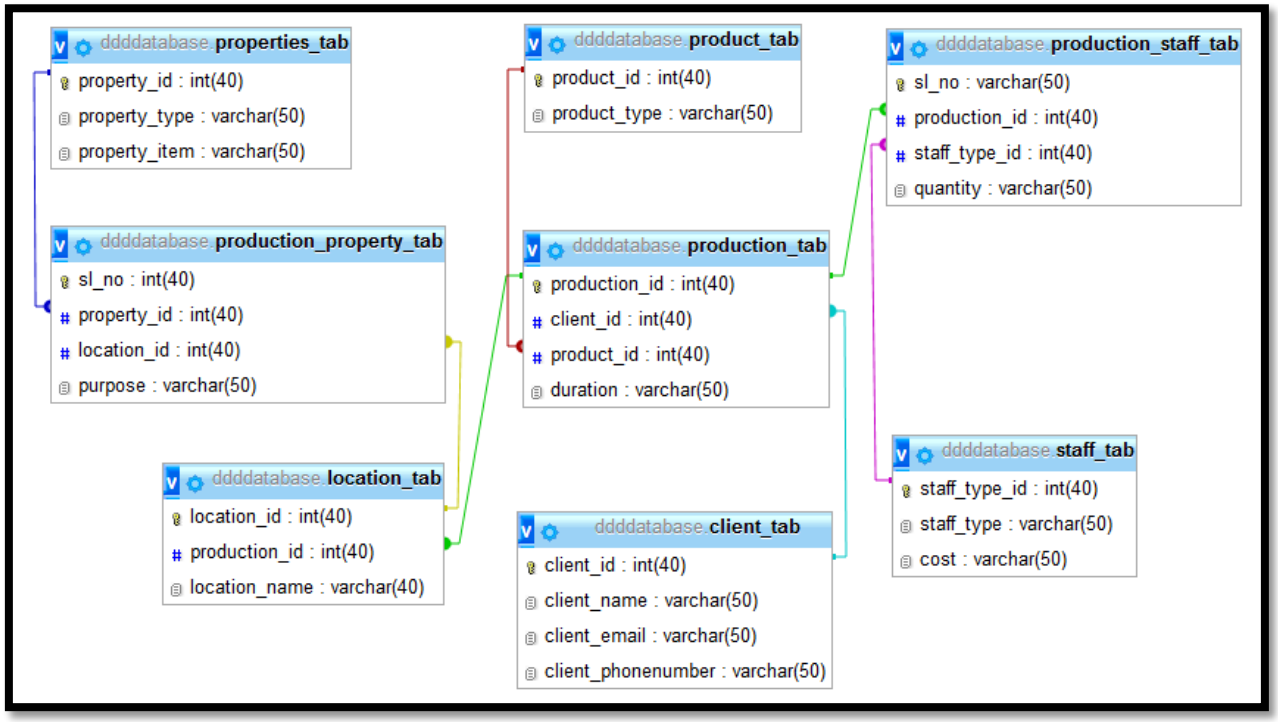


Fig. 1

(b) Show Normalizations Process:

Normalization:

This process of moving from data is not in a relation from, to a relation, and finally to a set of ideal relation is known as normalization. There are ***three type of normalization*** from. Including this:

- ***1st Normalization***
- ***2nd Normalization and***
- ***3rd Normalization***

Here I have described all three normalization. Flow this:

- **1st Normalization:**
  - 1st normalization means data will be ***without duplication***.
  - Because duplication data is big problem for database system. Such as ***wastes space*** and ***compromising data integrity.***
  - This database does not use any duplication data. We know that this process is 1st normalization.

For Example:

+ Options

	sl_no	production_id	staff_type_id	quantity
Edit Copy Delete	1	2	1	2
Edit Copy Delete	10	7	2	1
Edit Copy Delete	11	7	5	1
Edit Copy Delete	12	7	4	1
Edit Copy Delete	2	2	2	1
Edit Copy Delete	3	2	3	3
Edit Copy Delete	4	2	5	1
Edit Copy Delete	5	6	1	2
Edit Copy Delete	6	6	2	1
Edit Copy Delete	7	6	5	1
Edit Copy Delete	8	6	4	5
Edit Copy Delete	9	7	1	2

Fig. 2

- **2nd Normalization:**
  - 2nd normalization means database will be ***primary and foreign key***.
  - So this data base is 2nd normalization from database.
  - Because this database have primary and foreign key.

For Example:

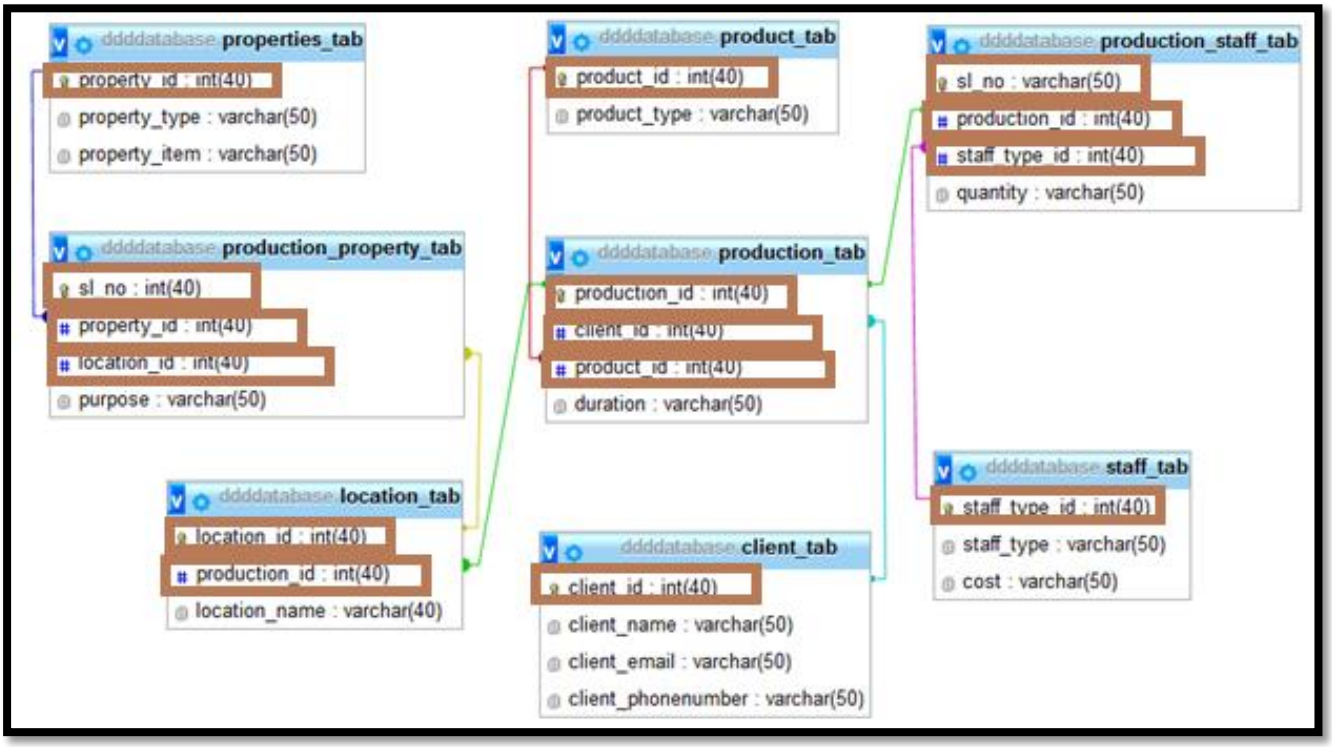


Fig. 3

- **3rd Normalization:**
  - ***3rd normalization*** means database will be ***without data dependency***.
  - This database is 3rd normalization from because this data base ***does not use data dependency***.

(Connolly, n.d.)

For Example:

+ Options

	production_id	client_id	product_id	duration
Edit Copy Delete	2	1	1	5
Edit Copy Delete	6	2	2	1
Edit Copy Delete	7	2	3	2

Fig. 4

+ Options

	client_id	client_name	client_email	client_phonenumber
Edit Copy Delete	1	Epom Motors	e_pom@gmail.com	+19293652546
Edit Copy Delete	2	Ministry ofAgriculture,Fisheriesand Food	ministry_agriculture@gmail.com	+19293652547

Fig. 5

(c) Data Dictionary for Entity relationship Model:

Entity Name	Attributes	Length	Key	Type
Client_tab	client_id	40	PRIMARY KEY	int
	client_name	50	NOT NULL	varchar
	cemail_email	50	NOT NULL	varchar
	client_phonenumber	50	NOT NULL	varchar
location_tab	location_id	40	PRIMARY KEY	int
	production_id	40	FOREIGN KEY	int
	location_name	40	NOT NULL	varchar
production_property_tab	sl_no	40	PRIMARY KEY	int
	property_id	40	FOREIGN KEY	int
	location_id	40	FOREIGN KEY	int
	purpose	50	NOT NULL	varchar
production_staff_tab	sl_no	50	PRIMARY KEY	varchar
	production_id	40	FOREIGN KEY	int
	staff_type_id	40	FOREIGN KEY	int
	quentity	50	NOT NULL	varchar
production_tab	production_id	40	PRIMARY KEY	int
	client_id	40	FOREIGN KEY	int
	product_id	40	FOREIGN KEY	int
	duration	50	NOT NULL	varchar
product_tab	product_id	40	PRIMARY KEY	int
	product_type	50	NOT NULL	varchar
properties_tab	property_id	40	PRIMARY KEY	int
	property_type	50	NOT NULL	varchar
	property_item	50	NOT NULL	varchar
staff_tab	staff_type_id	40	PRIMARY KEY	int
	staff_type	50	NOT NULL	varchar
	cost	50	NOT NULL	varchar

Fig. 6

Task-2

Data and Queries:

(a) Data for all production, customer and staff type:

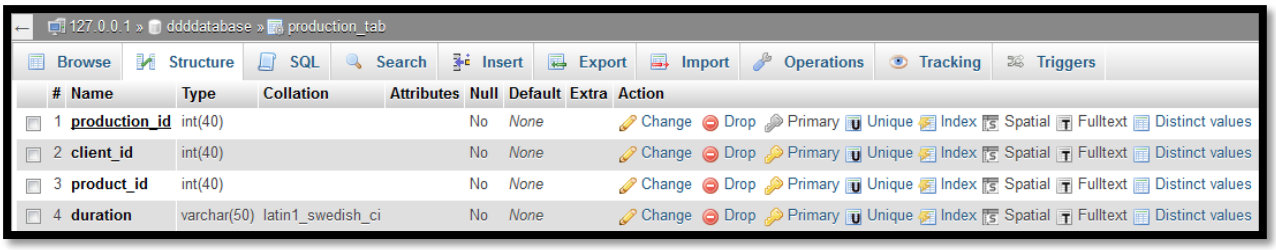
"production_tab" Table:

- Alter Table code for ***"production_tab"***

```
ALTER TABLE `production_tab`
  ADD CONSTRAINT `production_tab_ibfk_1` FOREIGN KEY (`client_id`) REFERENCES `client_tab` (`client_id`),
  ADD CONSTRAINT `production_tab_ibfk_2` FOREIGN KEY (`product_id`) REFERENCES `product_tab` (`product_id`);
```

Fig. 7

- Table for ***"production_tab"***



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	production_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	client_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	product_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
4	duration	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Fig. 8

- ***Insert*** values for ***"production_tab"***

```
INSERT INTO `production_tab` (`production_id`, `client_id`, `product_id`, `duration`) VALUES
(2, 1, 1, '5'),
(6, 2, 2, '1'),
(7, 2, 3, '2');
```

Fig. 9

- ***Output*** values for ***"production_tab"***

+ Options

	production_id	client_id	product_id	duration
Edit Copy Delete	2	1	1	5
Edit Copy Delete	6	2	2	1
Edit Copy Delete	7	2	3	2

Fig. 10

- Code for ***"production_tab"***

```
CREATE TABLE IF NOT EXISTS `production_tab` (
  `production_id` int(40) NOT NULL,
  `client_id` int(40) NOT NULL,
  `product_id` int(40) NOT NULL,
  `duration` varchar(50) NOT NULL,
  PRIMARY KEY (`production_id`),
  KEY `client_id` (`client_id`),
  KEY `product_id` (`product_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 11

"client_tab" Table:

- Table for ***"client_tab"***

127.0.0.1 » ddddatabase » client_tab

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Tracking | Triggers

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	client_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
2	client_name	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext More
3	client_email	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext More
4	client_phonenumber	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext More

Fig. 12

- ***Insert*** values for ***"client_tab"***

```
INSERT INTO `client_tab` (`client_id`, `client_name`, `client_email`, `client_phonenumber`) VALUES
(1, 'Epom Motors', 'e_pom@gmail.com', '+19293652546'),
(2, 'Ministry ofAgriculture,Fisheriesand Food', 'ministry_agriculture@gmail.com', '+19293652547');
```

Fig. 13

- ***Output*** values for ***"client_tab"***

+ Options

	client_id	client_name	client_email	client_phonenumber
Edit Copy Delete	1	Epom Motors	e_pom@gmail.com	+19293652546
Edit Copy Delete	2	Ministry ofAgriculture,Fisheriesand Food	ministry_agriculture@gmail.com	+19293652547

Fig. 14

- Code for ***"client_tab"***

```
CREATE TABLE IF NOT EXISTS `client_tab` (
  `client_id` int(40) NOT NULL,
  `client_name` varchar(50) NOT NULL,
  `client_email` varchar(50) NOT NULL,
  `client_phonenumber` varchar(50) NOT NULL,
  PRIMARY KEY (`client_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 15

"staff_tab" Table:

- Table for ***"staff_tab"***

127.0.0.1 » ddddatabase » staff_tab

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Tracking | Triggers

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	staff_type_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	staff_type	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	cost	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Fig. 16

- ***Insert*** values for ***"staff_tab"***

```
INSERT INTO `staff_tab` (`staff_type_id`, `staff_type`, `cost`) VALUES
(1, 'Camera Crew', '100'),
(2, 'Runner', '25'),
(3, 'Actor', '200'),
(4, 'Voice Actor', '100'),
(5, 'Producer', '550');
```

Fig. 17

- ***Output*** values for ***"staff_tab"***

+ Options

			staff_type_id	staff_type	cost
Edit	Copy	Delete	1	Camera Crew	100
Edit	Copy	Delete	2	Runner	25
Edit	Copy	Delete	3	Actor	200
Edit	Copy	Delete	4	Voice Actor	100
Edit	Copy	Delete	5	Producer	550

Fig. 18

- code for ***"staff_tab"***

```
CREATE TABLE IF NOT EXISTS `staff_tab` (
  `staff_type_id` int(40) NOT NULL,
  `staff_type` varchar(50) NOT NULL,
  `cost` varchar(50) NOT NULL,
  PRIMARY KEY (`staff_type_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 19

(b) Data for location:

"location_tab" Table:

- ***Alter*** table code for ***"location_tab"***

```
ALTER TABLE `location_tab`
  ADD CONSTRAINT `location_tab_ibfk_1` FOREIGN KEY (`production_id`) REFERENCES `production_tab` (`production_id`);
```

Fig. 20

- Table for ***"location_tab"***

127.0.0.1 » ddddatabase » location_tab

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Tracking | Triggers

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	location_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	production_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	location_name	varchar(40)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Fig. 21

- ***Insert*** values for ***"location_tab"***

```
INSERT INTO `location_tab` (`location_id`, `production_id`, `location_name`) VALUES
(1, 2, 'Millwall Park, Isleof Dogs, London'),
(2, 2, 'Windsor Castle, Grounds'),
(3, 6, 'Orford Ness, Suffolk'),
(4, 6, 'Rancid Attic Studio'),
(5, 7, 'St Jamess Park, London');
```

Fig. 22

- ***Output*** values for ***"location_tab"***

+ Options

	location_id	production_id	location_name
Edit Copy Delete	1	2	Millwall Park, Isleof Dogs, London
Edit Copy Delete	2	2	Windsor Castle, Grounds
Edit Copy Delete	3	6	Orford Ness, Suffolk
Edit Copy Delete	4	6	Rancid Attic Studio
Edit Copy Delete	5	7	St Jamess Park, London

Fig. 23

- Code for ***"location_tab"***

```
CREATE TABLE IF NOT EXISTS `location_tab` (
  `location_id` int(40) NOT NULL,
  `production_id` int(40) NOT NULL,
  `location_name` varchar(40) NOT NULL,
  PRIMARY KEY (`location_id`),
  KEY `production_id` (`production_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 24

"product_tab" Table:

- Table for ***"product_tab"***

127.0.0.1 » ddddatabase » product_tab

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Tracking | Triggers

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	product_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	product_type	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Fig. 25

- ***Insert*** values for ***"product_tab"***

```
INSERT INTO `product_tab` (`product_id`, `product_type`) VALUES
(1, 'Advertisement'),
(2, 'InformationFilm'),
(3, 'Training Film');
```

Fig. 26

- ***output*** values for ***"product_tab"***

+ Options

←T→	product_id	product_type
Edit Copy Delete	1	Advertisement
Edit Copy Delete	2	InformationFilm
Edit Copy Delete	3	Training Film

Fig. 27

- code for ***"product_tab"***

```
CREATE TABLE IF NOT EXISTS `product_tab` (
  `product_id` int(40) NOT NULL,
  `product_type` varchar(50) NOT NULL,
  PRIMARY KEY (`product_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 28

(c) Data on the properties and the location they are used at on a production: "properties_tab" Table:

- Table for ***"properties_tab"***

127.0.0.1 » ddddatabase » properties_tab

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Tracking | Triggers

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	property_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	property_type	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	property_item	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Fig. 29

- ***Insert*** values for ***"properties_tab"***

```
INSERT INTO `properties_tab` (`property_id`, `property_type`, `property_item`) VALUES
(1, 'Vehicle', 'Car'),
(2, 'Vehicle', 'Tractor'),
(3, 'Vehicle', 'Boat'),
(4, 'Furniture', 'Chair'),
(5, 'Furniture', 'Table'),
(6, 'Building', 'Suburban House'),
(7, 'Building', 'Inner City House');
```

Fig. 30

- ***Output*** values for ***"properties_tab"***

+ Options

←T→	property_id	property_type	property_item
Edit Copy Delete	1	Vehicle	Car
Edit Copy Delete	2	Vehicle	Tractor
Edit Copy Delete	3	Vehicle	Boat
Edit Copy Delete	4	Furniture	Chair
Edit Copy Delete	5	Furniture	Table
Edit Copy Delete	6	Building	Suburban House
Edit Copy Delete	7	Building	Inner City House

Fig. 31

- Code for ***"properties_tab"***

```
CREATE TABLE IF NOT EXISTS `properties_tab` (
  `property_id` int(40) NOT NULL,
  `property_type` varchar(50) NOT NULL,
  `property_item` varchar(50) NOT NULL,
  PRIMARY KEY (`property_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 32

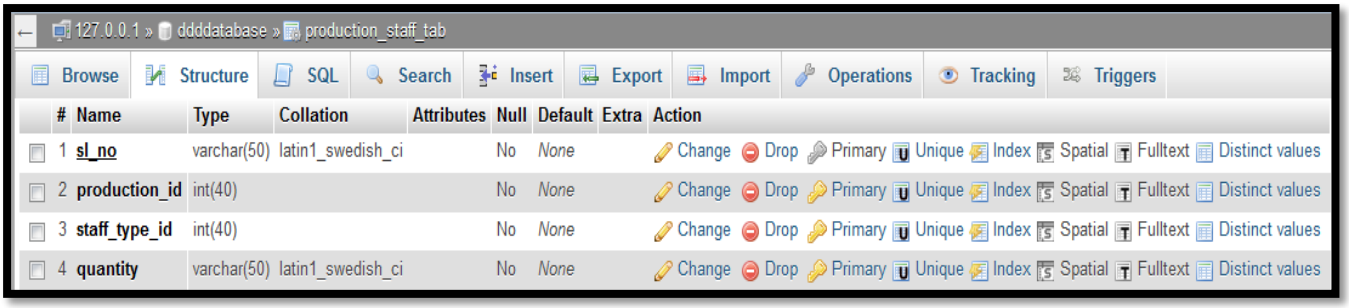
"production_staff_tab" Table:

- ***Alter*** Table code for ***"production_staff_tab"***

```
ALTER TABLE `production_staff_tab`
  ADD CONSTRAINT `production_staff_tab_ibfk_1` FOREIGN KEY (`production_id`) REFERENCES `production_tab` (`production_id`),
  ADD CONSTRAINT `production_staff_tab_ibfk_2` FOREIGN KEY (`staff_type_id`) REFERENCES `staff_tab` (`staff_type_id`);
```

Fig. 33

- Table for ***"production_staff_tab"***



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	sl_no	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	production_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	staff_type_id	int(40)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
4	quantity	varchar(50)	latin1_swedish_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Fig. 34

- ***Insert*** values for ***"production_staff_tab"***

```
INSERT INTO `production_staff_tab` (`sl_no`, `production_id`, `staff_type_id`, `quantity`) VALUES
('1', 2, 1, '2'),
('10', 7, 2, '1'),
('11', 7, 5, '1'),
('12', 7, 4, '1'),
('2', 2, 2, '1'),
('3', 2, 3, '3'),
('4', 2, 5, '1'),
('5', 6, 1, '2'),
('6', 6, 2, '1'),
('7', 6, 5, '1'),
('8', 6, 4, '1'),
('9', 7, 1, '2');
```

Fig. 35

- ***Output*** values for ***"production_staff_tab"***

+ Options

←T→	sl_no	production_id	staff_type_id	quantity
Edit Copy Delete	1	2	1	2
Edit Copy Delete	10	7	2	1
Edit Copy Delete	11	7	5	1
Edit Copy Delete	12	7	4	1
Edit Copy Delete	2	2	2	1
Edit Copy Delete	3	2	3	3
Edit Copy Delete	4	2	5	1
Edit Copy Delete	5	6	1	2
Edit Copy Delete	6	6	2	1
Edit Copy Delete	7	6	5	1
Edit Copy Delete	8	6	4	1
Edit Copy Delete	9	7	1	2

Fig. 36

- Code for ***"production_staff_tab"***

```
CREATE TABLE IF NOT EXISTS `production_staff_tab` (
  `sl_no` varchar(50) NOT NULL,
  `production_id` int(40) NOT NULL,
  `staff_type_id` int(40) NOT NULL,
  `quantity` varchar(50) NOT NULL,
  PRIMARY KEY (`sl_no`),
  KEY `production_id` (`production_id`),
  KEY `staff_type_id` (`staff_type_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 37

(d) Data for property types:

"production_property_tab" Table:

- ***Alter*** Table code for ***"production_property_tab"***

```
ALTER TABLE `production_property_tab`
  ADD CONSTRAINT `production_property_tab_ibfk_1` FOREIGN KEY (`property_id`) REFERENCES `properties_tab` (`property_id`),
  ADD CONSTRAINT `production_property_tab_ibfk_2` FOREIGN KEY (`location_id`) REFERENCES `location_tab` (`location_id`);
```

Fig. 38

- Table for ***"production_property_tab"***

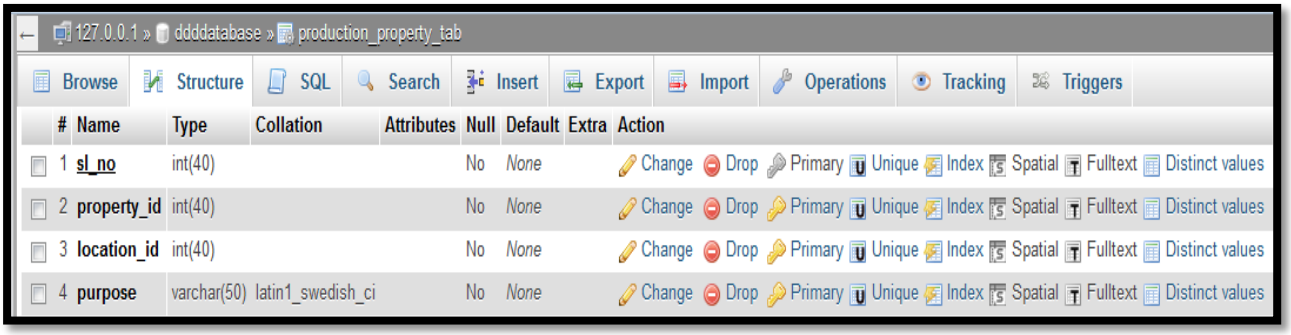


Fig. 39

- ***Insert*** values for ***"production_property_tab"***

```
INSERT INTO `production_property_tab` (`sl_no`, `property_id`, `location_id`, `purpose`) VALUES
(1, 1, 1, 'Wrecked'),
(2, 1, 2, 'Rent'),
(3, 1, 3, 'Rent'),
(4, 2, 5, 'Rent');
```

Fig. 40

- ***Output*** values for ***"production_property_tab"***

+ Options

	sl_no	property_id	location_id	purpose
Edit Copy Delete	1	1	1	Wrecked
Edit Copy Delete	2	1	2	Rent
Edit Copy Delete	3	1	3	Rent
Edit Copy Delete	4	2	5	Rent

Fig. 41

- Code for ***"production_property_tab"***

```
CREATE TABLE IF NOT EXISTS `production_property_tab` (
  `sl_no` int(40) NOT NULL,
  `property_id` int(40) NOT NULL,
  `location_id` int(40) NOT NULL,
  `purpose` varchar(50) NOT NULL,
  PRIMARY KEY (`sl_no`),
  KEY `property_id` (`property_id`),
  KEY `location_id` (`location_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 42

(Palgrave, n.d.)

(e) Query for selected all the production and which staff type have worked:

- Screen short for query ***code***

```
SELECT `production_id`,staff_type,`quantity`
FROM production_staff_tab ps, staff_tab st
WHERE ps.staff_type_id=st.staff_type_id
```

Fig. 43

- ***Run*** successfully

Showing rows 0 - 11 (12 total, Query took 0.0000 sec)

```
SELECT `production_id` , staff_type, `quantity`
FROM production_staff_tab ps, staff_tab st
WHERE ps.staff_type_id = st.staff_type_id
LIMIT 0 , 30
```

Fig. 44

- ***Output*** values

+ Options

production_id	staff_type	quantity
2	Camera Crew	2
6	Camera Crew	2
7	Camera Crew	2
7	Runner	1
2	Runner	1
6	Runner	1
2	Actor	3
7	Voice Actor	1
6	Voice Actor	1
7	Producer	1
2	Producer	1
6	Producer	1

Fig. 45

(f) Query for selected the properties that are actually used on a production:

- Screen short for query ***code***

```
SELECT pt.property_id,`property_type`,`property_item`
FROM `properties_tab` pt,production_property_tab pp
where pt.property_id=pp.property_id
```

Fig. 46

- ***Run*** successfully

Showing rows 0 - 3 (4 total, Query took 0.0000 sec)

```
SELECT pt.property_id, `property_type` , `property_item`
FROM `properties_tab` pt, production_property_tab pp
WHERE pt.property_id = pp.property_id
LIMIT 0 , 30
```

Fig. 47

- ***Output*** values

+ Options

	property_id	property_type	property_item
Edit Copy Delete	2	Vehicle	Tractor
Edit Copy Delete	2	Vehicle	Tractor
Edit Copy Delete	2	Vehicle	Tractor
Edit Copy Delete	2	Vehicle	Tractor

Fig. 48

(g) Query for selected all the location and the production that have taken place at those location:

- Screen short for query ***code***

```
SELECT lt.location_id,lt.production_id,lt.location_name
FROM location_tab lt,production_property_tab pp
where lt.location_id=pp.location_id
```

Fig. 49

- ***Run*** successfully

Showing rows 0 - 3 (4 total, Query took 0.0000 sec)

```
SELECT lt.location_id, lt.production_id, lt.location_name
FROM location_tab lt, production_property_tab pp
WHERE lt.location_id = pp.location_id
LIMIT 0 , 30
```

Fig. 50

- ***Output*** values

+ Options

←T→	location_id	production_id	location_name
Edit Copy Delete	1	2	Millwall Park, Isleof Dogs, London
Edit Copy Delete	2	2	Windsor Castle, Grounds
Edit Copy Delete	3	6	Orford Ness, Suffolk
Edit Copy Delete	5	7	St Jamess Park, London

Fig. 51

(h) Query for show the productions, location and properties:

- Screen short for query ***code***

```
SELECT lt.location_id,lt.production_id,lt.location_name,pp.property_id
FROM location_tab lt,production_property_tab pp
where lt.location_id=pp.location_id and lt.production_id=2
```

Fig. 52

- ***Run*** successfully

Showing rows 0 - 1 (2 total, Query took 0.0000 sec)

```
SELECT lt.location_id, lt.production_id, lt.location_name, pp.property_id
FROM location_tab lt, production_property_tab pp
WHERE lt.location_id = pp.location_id
AND lt.production_id =2
LIMIT 0 , 30
```

Fig. 53

- ***Output*** values

+ Options

location_id	production_id	location_name	property_id
1	2	Millwall Park, Isleof Dogs, London	1
2	2	Windsor Castle, Grounds	1

Fig. 54

(i) Quire count of number of location for production:

- Screen short for query ***code***

```
SELECT count(production_id)
FROM `location_tab`
WHERE production_id=2
```

Fig. 55

- ***Run*** successfully

Your SQL query has been executed successfully

```
SELECT COUNT( production_id )
FROM `location_tab`
WHERE production_id =2
```

Fig. 56

- ***Output*** values

+ Options

count(production_id)
2

Fig. 57

(j) Update the daily rate pay for runner:

- Update for ***"staff_tab"*** table
- Here update ***cost-30*** where "***staff-type_id=2"***
- Table for before update

+ Options

			staff_type_id	staff_type	cost
Edit	Copy	Delete	1	Camera Crew	100
Edit	Copy	Delete	2	Runner	25
Edit	Copy	Delete	3	Actor	200
Edit	Copy	Delete	4	Voice Actor	100
Edit	Copy	Delete	5	Producer	550

Fig. 58

- Screen short for query ***code***

```
UPDATE staff_tab SET cost=30
WHERE staff_type_id=2;
```

Fig. 59

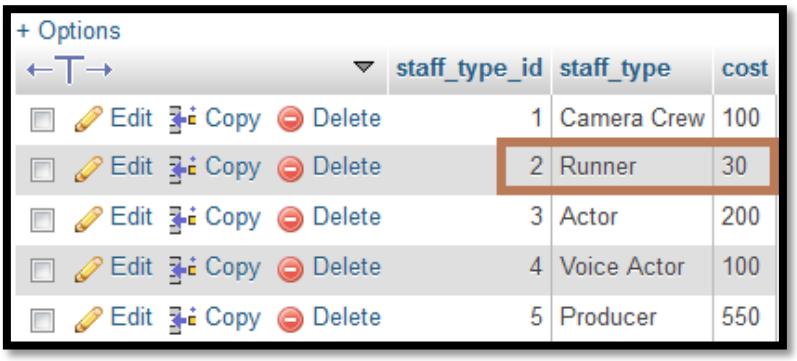
- ***Run*** successfully

1 row affected. (Query took 0.0710 sec)

```
UPDATE staff_tab SET cost =30 WHERE staff_type_id =2
```

Fig. 60

- Screen short for ***after update***



+ Options

				staff_type_id	staff_type	cost
Edit	Copy	Delete		1	Camera Crew	100
Edit	Copy	Delete		2	Runner	30
Edit	Copy	Delete		3	Actor	200
Edit	Copy	Delete		4	Voice Actor	100
Edit	Copy	Delete		5	Producer	550

Fig. 61

(k) Update name of "Epom Motors" to Epom Vehicle Management:

- Update for ***"client_tab"*** table
- Here update ***"client_name" "Epom Vehicale Management"*** where ***client_id=1***
- Table for ***before update***

+ Options

	client_id	client_name	client_email	client_phonenumber
Edit Copy Delete	1	Epom Motors	e_pom@gmail.com	+19293652546
Edit Copy Delete	2	Ministry ofAgriculture,Fisheriesand Food	ministry_agriculture@gmail.com	+19293652547

Fig. 62

- Screen short for query ***code***

```
1 UPDATE client_tab SET client_name='Epom Vehicle Management'
2 WHERE client_id=1;
```

Fig. 63

- ***Run*** successfully

1 row affected. (Query took 0.0800 sec)

```
UPDATE client_tab SET client_name = 'Epom Vehicle Management' WHERE client_id =1
```

Fig. 64

- Screen short for ***after update***

+ Options

	client_id	client_name	client_email	client_phonenumber
Edit Copy Delete	1	Epom Vehicle Management	e_pom@gmail.com	+19293652546
Edit Copy Delete	2	Ministry ofAgriculture,Fisheriesand Food	ministry_agriculture@gmail.com	+19293652547

Fig. 65

(I) Update record for five actor:

- Update for ***"production_staff_tab"*** table
- Here update ***"quantity=5"*** where ***production_id=6*** and ***"staff_type_id=1"***
- Table for ***before update***

+ Options

	sl_no	production_id	staff_type_id	quantity
Edit Copy Delete	1	2	1	2
Edit Copy Delete	10	7	2	1
Edit Copy Delete	11	7	5	1
Edit Copy Delete	12	7	4	1
Edit Copy Delete	2	2	2	1
Edit Copy Delete	3	2	3	3
Edit Copy Delete	4	2	5	1
Edit Copy Delete	5	6	1	2
Edit Copy Delete	6	6	2	1
Edit Copy Delete	7	6	5	1
Edit Copy Delete	8	6	4	1
Edit Copy Delete	9	7	1	2

Fig. 66

- Screen short for query ***code***

```
UPDATE production_staff_tab SET quantity=5
WHERE production_id=6 AND staff_type_id=4;
```

Fig. 67

- ***Run*** successfully

1 row affected. (Query took 0.0500 sec)

```
UPDATE production_staff_tab SET quantity =5 WHERE production_id =6 AND staff_type_id =4
```

Fig. 68

- Screen short for ***after update***

+ Options

	sl_no	production_id	staff_type_id	quantity
Edit Copy Delete	1	2	1	2
Edit Copy Delete	10	7	2	1
Edit Copy Delete	11	7	5	1
Edit Copy Delete	12	7	4	1
Edit Copy Delete	2	2	2	1
Edit Copy Delete	3	2	3	3
Edit Copy Delete	4	2	5	1
Edit Copy Delete	5	6	1	2
Edit Copy Delete	6	6	2	1
Edit Copy Delete	7	6	5	1
Edit Copy Delete	8	6	4	5
Edit Copy Delete	9	7	1	2

Fig. 69

(m) Add new staff type "Digital Effect supervisor":

- Add for ***"staff_tab"*** table
- Here add ***"staff_type"*** and ***cost*** where values ***"staff_type" "Digital Effect supervisor",*** and ***cost 250***
- Table for ***before Add***

+ Options

	staff_type_id	staff_type	cost
Edit Copy Delete	1	Camera Crew	100
Edit Copy Delete	2	Runner	30
Edit Copy Delete	3	Actor	200
Edit Copy Delete	4	Voice Actor	100
Edit Copy Delete	5	Producer	550

Fig. 70

- Screen short for query ***code***

```
INSERT INTO  staff_tab  (staff_type,  cost)
VALUES ('Digital Staff Supervisor', 250);
```

Fig. 71

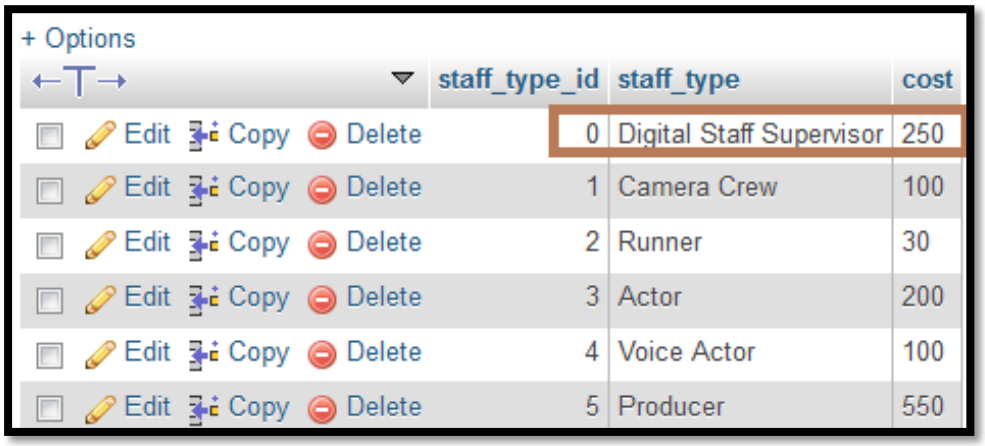
- ***Run*** successfully

1 row inserted. (Query took 0.0440 sec)

```
INSERT INTO staff_tab( staff_type, cost )
VALUES (
  'Digital Staff Supervisor', 250
)
```

Fig. 72

- Screen short for ***after Add***



+ Options

	staff_type_id	staff_type	cost
Edit Copy Delete	0	Digital Staff Supervisor	250
Edit Copy Delete	1	Camera Crew	100
Edit Copy Delete	2	Runner	30
Edit Copy Delete	3	Actor	200
Edit Copy Delete	4	Voice Actor	100
Edit Copy Delete	5	Producer	550

Fig. 73

(Connolly, 2005)

Task-3

Derived Data:

- This company will be extending their database.
- So that the company needs Alter the ***"properties_tab"*** table for including property cost and they also needs ***"production_staff_tab"," production_property_tab"table***.
- The company need to add ***rent and wrecked cost*** for ***"properties_tab".***
- Then adding cost on ***properties_tab*** than the company can produce ***costing report*** for ***location, properties and staff types*.**

Implementing Methods:

"properties_tab" table:

- Code for ***Alter table***

```
ALTER TABLE properties_tab ADD rentCost decimal(19,2);
ALTER TABLE properties_tab ADD wreckedCost decimal(19,2);
```

Fig. 74

- ***Run*** successfully

```
ALTER TABLE properties_tab ADD rentCost decimal(19,2);# 7 rows affected.

ALTER TABLE properties_tab ADD wreckedCost decimal(19,2);# 7 rows affected.
```

Fig. 75

- Table for after Alter and update ***rentCost*** and ***wreckdCost***

+ Options

	property_id	property_type	property_item	rentCost	wreckedCost
Edit Copy Delete	1	Vehicle	Car	NULL	NULL
Edit Copy Delete	2	Vehicle	Tractor	NULL	NULL
Edit Copy Delete	3	Vehicle	Boat	NULL	NULL
Edit Copy Delete	4	Furniture	Chair	NULL	NULL
Edit Copy Delete	5	Furniture	Table	NULL	NULL
Edit Copy Delete	6	Building	Suburban House	NULL	NULL
Edit Copy Delete	7	Building	Inner City House	NULL	NULL

Fig. 76

- Screen short for ***update code***

```
UPDATE properties_tab SET rentCost = 100, wreckedCost =1000 WHERE property_id = 1;
UPDATE properties_tab SET rentCost = 150, wreckedCost =1500 WHERE property_id = 2;
UPDATE properties_tab SET rentCost = 100, wreckedCost =1000 WHERE property_id = 3;
UPDATE properties_tab SET rentCost = 10, wreckedCost =50 WHERE property_id = 4;
UPDATE properties_tab SET rentCost = 20, wreckedCost =70 WHERE property_id = 5;
UPDATE properties_tab SET rentCost = 1000, wreckedCost =10000 WHERE property_id = 6;
UPDATE properties_tab SET rentCost = 1000, wreckedCost =10000 WHERE property_id = 7;
```

Fig. 77

- ***Run*** successfully

```
UPDATE properties_tab SET rentCost = 100, wreckedCost =1000 WHERE property_id = 1;# MySQL returned an empty result set (i.e. zero rows).
UPDATE properties_tab SET rentCost = 150, wreckedCost =1500 WHERE property_id = 2;# MySQL returned an empty result set (i.e. zero rows).
UPDATE properties_tab SET rentCost = 100, wreckedCost =1000 WHERE property_id = 3;# MySQL returned an empty result set (i.e. zero rows).
UPDATE properties_tab SET rentCost = 10, wreckedCost =50 WHERE property_id = 4;# MySQL returned an empty result set (i.e. zero rows).
UPDATE properties_tab SET rentCost = 20, wreckedCost =70 WHERE property_id = 5;# MySQL returned an empty result set (i.e. zero rows).
UPDATE properties_tab SET rentCost = 1000, wreckedCost =10000 WHERE property_id = 6;# MySQL returned an empty result set (i.e. zero rows).
UPDATE properties_tab SET rentCost = 1000, wreckedCost =10000 WHERE property_id = 7;# MySQL returned an empty result set (i.e. zero rows).
```

Fig. 78

- After update ***rentCost*** and ***WreckCost.***

+ Options

	property_id	property_type	property_item	rentCost	wreckedCost
Edit Copy Delete	4	Furniture	Chair	10.00	50.00
Edit Copy Delete	5	Furniture	Table	20.00	70.00
Edit Copy Delete	1	Vehicle	Car	100.00	1000.00
Edit Copy Delete	3	Vehicle	Boat	100.00	1000.00
Edit Copy Delete	2	Vehicle	Tractor	150.00	1500.00
Edit Copy Delete	6	Building	Suburban House	1000.00	10000.00
Edit Copy Delete	7	Building	Inner City House	1000.00	10000.00

Fig. 79

- Code for ***all staff cost***

```
SELECT `sl_no`,`production_id`,pst.`staff_type_id`,quantity * cost
from production_staff_tab pst,staff_tab
where pst.staff_type_Id=staff_tab.staff_type_id
```

Fig. 80

- ***Run*** successfully

Showing rows 0 - 11 (12 total, Query took 0.0010 sec)

```
SELECT `sl_no` , `production_id` , pst.`staff_type_id` , quantity * cost
FROM production_staff_tab pst, staff_tab
WHERE pst.staff_type_id = staff_tab.staff_type_id
LIMIT 0 , 30
```

Fig. 81

- ***Output*** values

+ Options

sl_no	production_id	staff_type_id	quantity * cost
1	2	1	200
5	6	1	200
9	7	1	200
10	7	2	30
2	2	2	30
6	6	2	30
3	2	3	600
12	7	4	100
8	6	4	500
11	7	5	550
4	2	5	550
7	6	5	550

Fig. 82

- Staff code for ***particular production***

```
SELECT `sl_no`,`production_id`,pst.`staff_type_id`,quantity * cost
from production_staff_tab pst,staff_tab
where pst.staff_type_id=staff_tab.staff_type_id
and pst.production_id=2
```

Fig. 83

- ***Run*** successfully

Showing rows 0 - 3 (4 total, Query took 0.0000 sec)

```
SELECT `sl_no` , `production_id` , pst.`staff_type_id` , quantity * cost
FROM production_staff_tab pst, staff_tab
WHERE pst.staff_type_id = staff_tab.staff_type_id
AND pst.production_id =2
LIMIT 0 , 30
```

Fig. 84

- ***Output*** values

+ Options

sl_no	production_id	staff_type_id	quantity * cost
1	2	1	200
2	2	2	30
3	2	3	600
4	2	5	550

Fig. 85

- Code for ***total staff cost*** of production

```
SELECT sum(quantity * cost)
from production_staff_tab pst,staff_tab
where pst.staff_type_id=staff_tab.staff_type_id
and pst.production_id=2
```

Fig. 86

- ***Run*** successfully

Showing rows 0 - 0 (1 total, Query took 0.0000 sec)

```
SELECT SUM( quantity * cost )
FROM production_staff_tab pst, staff_tab
WHERE pst.staff_type_id = staff_tab.staff_type_id
AND pst.production_id =2
```

Fig. 87

- ***Output*** values

+ Options

sum(quantity * cost)
1380

Fig. 88

- Code for property ***rent cost***

```
SELECT `sl_no`,ppt.`property_id`,`location_id`,rentCost
from production_property_tab ppt,properties_tab pt
where ppt.property_id=pt.property_id
and `purpose`='Rent'
```

Fig. 89

- ***Run*** successfully

Showing rows 0 - 2 (3 total, Query took 0.0000 sec)

```
SELECT `sl_no` , ppt.`property_id` , `location_id` , rentCost
FROM production_property_tab ppt, properties_tab pt
WHERE ppt.property_id = pt.property_id
AND `purpose` = 'Rent'
LIMIT 0 , 30
```

Fig. 90

- ***Output*** values

+ Options

sl_no	property_id	location_id	rentCost
2	1	2	100.00
3	1	3	100.00
4	2	5	150.00

Fig. 91

- Code for total ***rent cost***

```
SELECT sum(rentCost)
from production_property_tab ppt,properties_tab pt
where ppt.property_id=pt.property_id
and `purpose`='rent'
```

Fig. 92

- ***Run*** successfully

```
Showing rows 0 - 0 ( 1 total, Query took 0.0000 sec)

SELECT SUM( rentCost )
FROM production_property_tab ppt, properties_tab pt
WHERE ppt.property_id = pt.property_id
AND `purpose` = 'rent'
```

Fig. 93

- ***Output*** values

```
+ Options
sum(rentCost)
350.00
```

Fig. 94

- Code for properties ***wrecked cost***

```
SELECT `sl_no`,ppt.`property_id`,`location_id`,wreckedCost
from production_property_tab ppt,properties_tab pt
where ppt.property_id=pt.property_id
and `purpose`='wrecked'
```

Fig. 95

- ***Run*** successfully

```
Showing rows 0 - 0 ( 1 total, Query took 0.0000 sec)

SELECT `sl_no` , ppt.`property_id` , `location_id` , wreckedCost
FROM production_property_tab ppt, properties_tab pt
WHERE ppt.property_id = pt.property_id
AND `purpose` = 'wrecked'
LIMIT 0 , 30
```

Fig. 96

- ***Output*** values

+ Options

sl_no	property_id	location_id	wreckedCost
1	1	1	1000.00

Fig. 97

- Code for total ***wrecked cost***

```
SELECT SUM(wreckedCost) As '[Total Wrecked Cost]'
from production_property_tab ppt,properties_tab pt
where ppt.property_id=pt.property_id
and `purpose`='wrecked'
```

Fig. 98

- ***Run*** successfully

Showing rows 0 - 0 (1 total, Query took 0.0000 sec)

```
SELECT SUM( wreckedCost ) AS '[Total Wrecked Cost]'
FROM production_property_tab ppt, properties_tab pt
WHERE ppt.property_id = pt.property_id
AND `purpose` = 'wrecked'
```

Fig. 99

- ***Output*** values

+ Options

[Total Wrecked Cost]
1000.00

Fig. 100

(Kroenke, (2007))

Task-4

Evaluation:

Assignment requirement:

- For this assignment I have ***used SQL server*** (My SQL version v3.2.1).
- I have access server used by ***local host***
- After I have created database name. my database name is ***"ddddatabase"***
- Then I have ***created 8 data table*** for ***store and keeping data***
- Then created data ***relationship model*** with ***primary and foreign key***.
- The all table are ***normalization*** from.
- After finish ***data dictionary*** table with values
- Then ***input query*** code. My query successfully runs.

Evaluation table including:

Task	Requirement	Performed Task	Remark
Task-1	Entity relationship model	Created ***entity relationship*** model	done
	Normalization process	***Fully 3rd normalization*** from show	
	Data Table	Completed ***data dictionary table***	
Task-2	Input Data	Input all data related of ***"production_tab", "client_tab"*** and ***"staff_tab" table***	
		Input data for ***"location_tab"*** and ***"product_tab"*** table	
		Input data for ***"properties_tab"*** and ***"production_staff_tab"*** table	
		Input data for ***"production_property_tab"*** table	
	Write a Query	Completed query are all the production which ***staff_type*** work them	
		Completed query for all the ***"properties_tab"*** which they used production	
		Completed query are all the ***"loction_tab"*** and	

		production which taken location	
	Write a Query	Completed query for all the ***"production_property_tab"*** and ***"location_tab"*** which taken particular production	done
		Completed query are that count number from ***"location_tab"*** for production	
	Update Query	Update ***"staff_tab"*** where update runner cost update 30 which ***staff_type_id=2***	
		Update ***"client_tab"*** where update ***'Epom Vehicle Management'*** which ***client_id=1***	
		Update ***"production_staff_tab"*** where update ***"quantity=5"*** which ***production_id=6*** and ***staff_type_id=1***	
	Add Query	Add ***"staff_tab"*** where update ***staff_type "Digital Effect supervisor",*** and ***cost 250***	
Task-3	Drive Data	Adding new cost information on table ***"properties_tab".*** Then all the ***cost information*** finds out including ***staff costing, property*** use ***costing***. Make a ***total sum*** of those costing on a ***particular production***.	

Conclusion:

I have fulfilled ***the entire requirement in this assignment.*** I have described all the ***process*** with ***screenshot*** and I have flowed ***database design***. ***And*** successfully run data query. I have ***gain a lot of knowledge*** in this assignment.

Bibliography

- Addision-Wesley, 2005. *Database Systems: A Practical Approach to Design and Implementation*. Fourth Edition ed. Addision-Wesley.
- Connolly, T.a.B.C., n.d. *Database Systems A Practical Approach to Design, Implementation and Management*. Fourth Edition ed. Addison and Wesley.
- Kroenke, D.M., (2007). *Database Concepts*. 3rd ed. Prentice Hall.
- Palgrave, n.d. *Database Systems*. Third Edition ed. Benyon-Davies.