

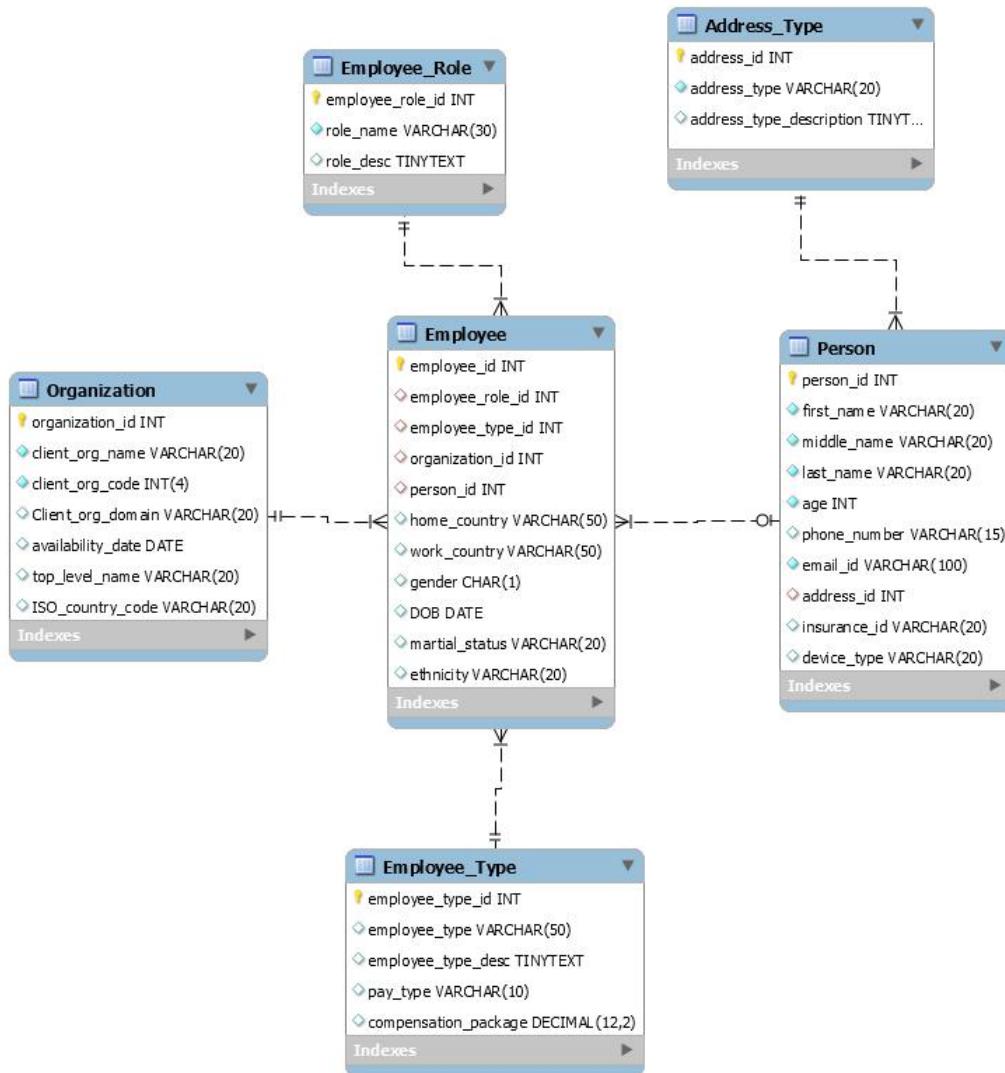
## Employee Schema

---

## Table of Contents

Employee Schema ER Diagram .....	3
Table Description .....	4
Explaining Queries .....	6
1. Query to create table .....	6
2. Query to create view .....	9
3. Query to create and call stored procedure .....	10
4. Query to inset data into tables.....	14
Best practices.....	16
Alternative methods .....	16
References: .....	16

## Employee Schema ER Diagram



The above-mentioned Schema defines the Employee entity and its relationships to other entities. Each employee is of a Person Type and each address of a person has an address\_type. The employee belongs to an organization and is of one employee Type.

## Table Description

### 1. Organization Table:

Fields	Type	Can_Be_Null?	Key	Extra
organization_id	INT(11)	NO	PRI	AUTO_INCREMENT
client_org_name	VARCHAR(20)	NO		
client_org_code	INT(4)	NO		
client_org_domain	VARCHAR(20)	YES		
availability_date	DATE	YES		
top_level_name	VARCHAR(20)	YES		
ISO_country_code	VARCHAR(20)	YES		

### 2. Employee Table:

Fields	Type	Can_Be_Null??	Key	Extra
employee_id	INT(11)	NO	PRI	AUTO_INCREMENT
employee_role_id	INT(11)	YES	MUL	
employee_type_id	INT(11)	YES	MUL	
organization_id	INT(11)	YES	MUL	
person_id	INT(11)	NO	MUL	
home_country	VARCHAR(50)	YES		
work_country	VARCHAR(50)	YES		
gender	CHAR(1)	YES		
DOB	DATE	YES		
marital_status	VARCHAR(20)	YES		
ethnicity	VARCHAR(20)			

3. Address\_Type Table:

Fields	Type	Can_Be_Null?	Key	Extra
address_id	INT(11)	NO	PRIMARY	AUTO_INCREMENT
address_type	VARCHAR(20)	NO		
address_type_description	VARCHAR(20)	YES		

4. Employee\_Type Table:

Fields	Type	CanBeNull ?	Key	Extra
employee_type_id	INT(11)	NO	PRI	AUTO_INCREMENT
employee_type	VARCHAR(50)	YES		
employee_type_desc	TINYTEXT	YES		
pay_type	VARCHAR(10)	YES		
Compensation_package	DECIMAL(12,2)	YES		

5. Person Table:

Fields	Type	CanBeNull ?	Key	Extra
person_id	INT(11)	NO	PRI	AUTO_INCREMENT
first_name	VARCHAR(20)	NO		
middle_name	VARCHAR(20)	NO		
last_name	VARCHAR(20)	NO		
age	INT(11)	NO		
phone_number	VARCHAR(15)	YES		
email_id	VARCHAR(20)	NO		
address_id	INT(11)	YES	MUL	
insurance_id	VARCHAR(20)	YES		
device_type	VARCHAR(20)	YES		

## 6. Employee\_Role Table:

Fields	Type	CanBeNull?	Key	Extra
employee_role_id	INT(11)	NO	PRI	AUTO_INCREMENT
role_name	VARCHAR(30)	NO		
Role_desc	TINYTEXT	YES		

## Explaining Queries

### 1. Query to create table

Executing the following query will create a Table named “Address\_Type” and add 3 columns to it namely address\_id, address\_type, address\_type.

```

1. -- Create table Address_Type --
2. CREATE TABLE IF NOT EXISTS Address_Type (
3.     address_id INT AUTO_INCREMENT PRIMARY KEY,
4.     address_type VARCHAR(20) NOT NULL,
5.     address_type_description TINYTEXT
6. );

```

	Field	Type	Null	Key	Default	Extra
▶	address_id	int(11)	NO	PRI	NULL	auto_increment
	address_type	varchar(20)	NO		NULL	
	address_type_description	tinytext	YES		NULL	

Similarly, other tables were also created, they are as follows:

```

1. -- Create table Employee_Type --
2. CREATE TABLE IF NOT EXISTS Employee_Type (
3.     employee_type_id INT AUTO_INCREMENT PRIMARY KEY,
4.     employee_type VARCHAR(50),
5.     employee_type_desc TINYTEXT,
6.     pay_type VARCHAR(10),
7.     compensation_package decimal(12,2)
8. );

```

	Field	Type	Null	Key	Default	Extra
▶	employee_role_id	int(11)	NO	PRI	NULL	
	role_name	varchar(30)	NO		NULL	
	role_desc	tinytext	YES		NULL	

```

1. -- Create table Employee_Role --
2. CREATE TABLE IF NOT EXISTS Employee_Role (
3.     employee_role_id INT PRIMARY KEY,
4.     role_name VARCHAR(30) NOT NULL,
5.     role_desc TINYTEXT
6. );

```

	Field	Type	Null	Key	Default	Extra
▶	employee_role_id	int(11)	NO	PRI	NULL	
	role_name	varchar(30)	NO		NULL	
	role_desc	tinytext	YES		NULL	

```

1. -- Create table Organization --
2. CREATE TABLE IF NOT EXISTS Organization(
3.     organization_id INT AUTO_INCREMENT PRIMARY KEY,
4.     client_org_name VARCHAR(20) NOT NULL,
5.     client_org_code INT(4) NOT NULL,
6.     Client_org_domain VARCHAR(20),
7.     availability_date DATE,
8.     top_level_name VARCHAR(20),
9.     ISO_country_code VARCHAR(20)
10. );

```

	Field	Type	Null	Key	Default	Extra
▶	organization_id	int(11)	NO	PRI	NULL	auto_increment
	Client_org_name	varchar(20)	YES		NULL	
	Client_org_code	int(4)	YES		NULL	
	Client_org_domain	varchar(20)	YES		NULL	
	availability_date	date	YES		NULL	
	top_level_name	varchar(20)	YES		NULL	
	ISO_country_code	varchar(20)	YES		NULL	

```

1. -- Create table Person --
2. CREATE TABLE IF NOT EXISTS Person(
3.     person_id INT AUTO_INCREMENT PRIMARY KEY,
4.     first_name VARCHAR(20) NOT NULL,
5.     middle_name VARCHAR(20) NOT NULL,
6.     last_name VARCHAR(20) NOT NULL,
7.     age INT NOT NULL,
8.     phone_number VARCHAR(15),
9.     email_id VARCHAR(100) NOT NULL,
10.    address_id INT,
11.    insurance_id VARCHAR(20),
12.    device_type VARCHAR(20),
13.        FOREIGN KEY (address_id) REFERENCES Address_Type (address_id)
14. );

```

	Field	Type	Null	Key	Default	Extra
▶	person_id	int(11)	NO	PRI	NULL	auto_increment
	first_name	varchar(20)	NO		NULL	
	middle_name	varchar(20)	NO		NULL	
	last_name	varchar(20)	NO		NULL	
	age	int(11)	NO		NULL	
	phone_number	varchar(15)	YES		NULL	
	email_id	varchar(50)	YES		NULL	
	address_id	int(11)	YES	MUL	NULL	
	insurance_id	varchar(20)	YES		NULL	
	device_type	varchar(20)	YES		NULL	

```

1. -- Create table Employee --
2. CREATE TABLE IF NOT EXISTS Employee (
3.     employee_id INT AUTO_INCREMENT PRIMARY KEY,
4.     employee_role_id INT ,
5.     employee_type_id INT ,
6.     organization_id INT ,
7.     person_id INT NOT NULL,
8.     home_country VARCHAR(50) ,
9.     work_country VARCHAR(50) ,
10.    gender CHAR(1), -- make a select list maybe
11.    DOB DATE, -- format type
12.    martial_status VARCHAR(20), -- letter based
13.    ethnicity VARCHAR(20),
14.    FOREIGN KEY(organization_id)
15.        REFERENCES Organization(organization_id),
16.    FOREIGN KEY(employee_role_id)

18.    FOREIGN KEY(person_id)

20.    FOREIGN KEY(employee_type_id)

22. );

```

	Field	Type	Null	Key	Default	Extra
▶	employee_id	int(11)	NO	PRI	NULL	auto_increment
	employee_role_id	int(11)	YES	MUL	NULL	
	employee_type_id	int(11)	YES	MUL	NULL	
	organization_id	int(11)	YES	MUL	NULL	
	person_id	int(11)	NO	MUL	NULL	
	home_country	varchar(50)	YES		NULL	
	work_country	varchar(50)	YES		NULL	
	gender	char(1)	YES		NULL	
	DOB	date	YES		NULL	
	martial_status	varchar(20)	YES		NULL	
	ethnicity	varchar(20)	YES		NULL	

## 2. Query to create view

The following view consists of 2 attributes from all the tables.

Select statement to view the data of employeeView

```
1. CREATE VIEW employeeview
2. AS
3. SELECT employee.employee_role_id as "Emp role id", person.first_name as
   "First name", person.middle_name as "Middle name",
4. employee_role.role_name "Emp Role name", employee_role.role_desc as "Emp
   Role description",
5. organization.Client_org_name as "Client Org name",
   employee_type.compensation_package as "Compensation
   Package", organization.top_level_name as "Top level name",
6. employee_type.employee_type as "Employee Type", person.age as "age", employee.DOB
   as "DOB", address_type.address_type as "Address Type",
7. address_type.address_type_description as "Address" , employee.home_country as
   "Home country"
8. FROM address_type, employee_type, employee_role, organization, person, employee
9. WHERE address_type.address_id = Person.address_id and
   employee_type.employee_type_id = employee.employee_type_id and
10. employee_role.employee_role_id = employee.employee_role_id and
    organization.organization_id = employee.organization_id and
11. person.person_id = employee.person_id order by employee.employee_role_id;
```

```
1. SELECT * FROM employeeView;
```

Screenshot of the view output:

Emp role id	First name	Middle name	Emp Role name	Emp Role description	Client Org name	Compensation Package	Top level name	Employee Type	age	DOB	Address Type	Address
3	Kerby	Pietro	Product Manager	Visionary zero defect migration	McD	0.05	level2	Engineering	34	1980-10-26	Home2	Integrated
4	Briana	Jeff	Product Manager	Robust hybrid frame	McD	0.03	level2	Training	43	1994-08-31	Home2	Integrated
4	Wadsworth	Wilbert	Product Manager	Robust hybrid frame	Boeing	0.00	level3	Business Development	33	1987-08-30	Commercial	Balanced
7	Torry	Raphael	Marketing Head	Fundamental system-worthy neural-net	McD	0.05	level2	Research and Development	46	1988-06-05	Emergency	Assimilate
8	Paulina	Pace	DBA specialist	Intuitive global help-desk	Boeing	0.03	level3	Product Management	32	1967-11-17	Commercial	Balanced
10	Elissa	Curry	Marketing Head	Programmable encompassing capacity	Chase	0.05	level4	Legal	53	1967-09-06	Commercial	Balanced
11	Mariissa	Raynard	QA Tester	Universal transitional frame	Chase	0.01	level1	Business Development	53	1976-01-17	Commercial	Balanced
13	Stormi	Chaddie	Marketing Head	Focused client-driven contingency	Chase	0.05	level2	Engineering	43	1987-10-02	Home2	Integrated
14	Paulina	Pace	QA Tester	Devolved holistic leverage	Boeing	0.01	level3	Support	32	1968-06-21	Commercial	Balanced
14	Jesse	Silvio	QA Tester	Devolved holistic leverage	McD	0.03	level2	Support	39	1977-09-11	Home2	Integrated
15	Kerhu	Pietro	DevOne	Multi-tiered non-volatile online	Boeing	0.01	level3	Marketing	34	1988-04-27	Home2	Integrated

Alternative method to create a view with Joins is given below:

```
1. CREATE VIEW employeeview_alternate
2. as
3. select e.employee_id as "Emp ID", p.first_name as "First name", p.last_name as
   "Last name", e.DOB as "DOB",
```

```

4. o.client_org_name as "Client org name", o.top_level_name as "Top Level name",
5. er.employee_role_id as "Emp Role id", er.role_name as "Role name", et.pay_type
   as "Pay type", et.compensation_package as "Compensation package",
6. a.address_id as "Address ID", a.address_type as "Address Type"
7. from address_type a
8. INNER join person p ON a.address_id=p.address_id
9. INNER join employee e ON e.person_id=p.person_id
10. inner join employee_role er ON e.employee_role_id=er.employee_role_id
11. inner join employee_type et ON e.employee_type_id=et.employee_type_id
12. inner join organization o on o.organization_id=e.organization_id order by
   e.employee_id;
13. Select statement to view the data of employeeView_alternate

```

1. SELECT \* FROM employeeView\_alternate;

	Emp ID	First name	Last name	DOB	Client org name	Top Level name	Emp Role id	Role name	Pay type	Compensation package	Address ID	Address Type
▶	1	Dyann	Windress	1998-07-05	Chase	level2	28	Sales	Monthly	0.30	2	Commercial
	2	Maurice	Kilmurray	1979-05-01	Chase	level1	22	DevOps	Bi-Weekly	0.05	2	Commercial
	3	Noel	Gooden	1983-12-05	McD	level2	46	DBA specialist	Bi-Weekly	0.02	2	Commercial
	4	Kinnie	Domsalla	1979-10-18	Chase	level1	20	Manager	Bi-Weekly	0.05	2	Commercial
	5	Mariissa	Tremblett	1976-01-17	Chase	level1	11	QA Tester	Hourly	0.01	2	Commercial
	6	Jesse	Woolcott	1977-09-11	McD	level2	14	QA Tester	Hourly	0.03	4	Home2

### 3. Query to create and call stored procedure

The following Stored Procedures can be called upon to insert data into the tables by passing values as parameters.

- Address\_Type

```

1. -- Stored procedure for inserting data into table Address_Type --
2. DELIMITER $$ 
3. CREATE PROCEDURE insert_address_type(
4. IN address_id INT ,
5. IN address_type VARCHAR(20) ,
6. IN address_type_description VARCHAR(20)
7. )
8. BEGIN
9.     INSERT INTO Address_Type VALUES (address_id,
   address_type, address_type_description);
10. END $$ 
11. DELIMITER ;

```

Result Grid			Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	address_id	address_type	address_type_description			
▶	1	Home2	Decentralized static approach			
	2	Commercial	Balanced user-facing flexibility			
	3	Emergency	Assimilated next generation Graphical User Inte...			
	4	Home2	Integrated content-based parallelism			
*	NULL	NULL	NULL			

- Employee

```

1. -- Stored procedure for inserting data into table Employee --
2. DELIMITER $$ 
3. CREATE PROCEDURE insert_employee(
4. IN employee_id INT ,
5. IN employee_role_id INT ,
6. IN employee_type_id INT ,
7. IN organization_id INT ,
8. IN person_id INT,
9. IN home_country VARCHAR(50),
10. IN work_country VARCHAR(50),
11. IN gender CHAR(1),
12. IN DOB DATE,
13. IN martial_status VARCHAR(20),
14. IN ethnicity VARCHAR(20)
15. )
16. BEGIN
17.     INSERT INTO Employee VALUES (employee_id, employee_role_id, employee_type_id,
organization_id, person_id, home_country, work_country, gender, DOB,
martial_status, ethnicity);
18. END $$ 
19. DELIMITER ;

```

The screenshot shows a MySQL Workbench interface with a result grid titled 'Result Grid'. The grid displays 19 rows of data from the Employee table. The columns are: employee\_id, employee\_role\_id, employee\_type\_id, organization\_id, person\_id, home\_country, work\_country, gender, DOB, martial\_status, and ethnicity. The data includes various countries like Costa Rica, China, Portugal, Mexico, etc., and roles like Single Parent, Married, Divorced, etc.

	employee_id	employee_role_id	employee_type_id	organization_id	person_id	home_country	work_country	gender	DOB	martial_status	ethnicity
▶	1	28	33	8	10	Costa Rica	Denmark	F	1998-07-05	Single Parent	Hispanic
	2	22	25	10	20	China	Costa Rica	M	1979-05-01	Married	White/American
	3	46	3	2	40	Portugal	Lithuania	F	1983-12-05	single	Hispanic
	4	20	16	10	2	Mexico	Portugal	F	1979-10-18	Divorced	South-Asian
	5	11	41	10	15	China	Peru	M	1976-01-17	Married	Hispanic
	6	14	2	3	43	Poland	Indonesia	F	1977-09-11	Single Parent	Hispanic
	7	3	19	2	30	Japan	France	F	1980-10-26	Married	South-Asian
	8	49	32	6	23	China	South Korea	M	1984-05-05	single	Hispanic
	9	37	33	9	7	China	Finland	M	1997-03-24	Single Parent	South-Asian
	10	33	47	6	10	Colombia	Ukraine	F	1988-10-15	single	South-Asian
	11	25	4	3	11	Indonesia	Indonesia	M	1973-01-10	Divorced	White/American
	12	32	47	6	31	Indonesia	Russia	M	1980-02-04	Single Parent	White/American
	13	15	29	7	30	China	Russia	M	1988-04-22	Divorced	White/American
	14	48	2	4	3	China	France	F	1976-07-14	Divorced	South-Asian
	15	32	13	9	27	Argentina	China	F	1977-02-10	Single Parent	Muslim
	16	34	31	3	48	China	Argentina	M	1990-10-28	Married	South-Asian
	17	33	34	3	16	China	China	F	1968-12-29	single	South-Asian
	18	42	34	6	46	Haiti	Malta	F	1976-08-17	single	Hispanic
	19	7	12	3	19	Russia	Thailand	F	1988-06-05	Divorced	Muslim

```

1. -- Stored procedure for inserting data into table Employee_Role --
2. DELIMITER $$ 
3. CREATE PROCEDURE insert_employee_role(
4. IN employee_role_id INT,
5. IN role_name VARCHAR(30),
6. IN role_desc TINYTEXT
7. )
8. BEGIN
9.     INSERT INTO Employee_Role VALUES (employee_role_id, role_name,role_desc);
10. END $$ 
11. DELIMITER ;

```

Result Grid | Filter Rows:  | Edit: Export/Import:

	employee_role_id	role_name	role_desc
▶	1	Product Manager	Decentralized systemic productivity
	2	Receptionist	Total fresh-thinking policy
	3	Product Manager	Visionary zero defect migration
	4	Product Manager	Robust hybrid frame
	5	Receptionist	User-centric multi-state productivity
	6	Engineer	Managed systematic intranet
	7	Marketing Head	Fundamental system-worthy neural-net
	8	DBA specialist	Intuitive global help-desk
	9	Sales	Organic object-oriented superstructure
	10	Marketing Head	Programmable encompassing capacity
	11	QA Tester	Universal transitional frame
	12	CTO	Object-based executive algorithm
	13	Marketing Head	Focused client-driven contingency
	14	QA Tester	Devolved holistic leverage
	15	DevOps	Multi-tiered non-volatile policy
	16	DBA specialist	Programmable disintermediate moratori...
	17	CEO	Synchronised 24/7 functionalities
	18	Receptionist	Inverse even-keeled policy
	19	QA Tester	Universal even-keeled conglomeration

```

1. -- Stored procedure for inserting data into table Employee_Type --
2. DELIMITER $$ 
3. CREATE PROCEDURE insert_employee_type(
4. IN employee_type_id INT,
5. IN employee_type VARCHAR(50),
6. IN employee_type_desc TINYTEXT,
7. IN pay_type VARCHAR(10),
8. IN compensation_package decimal(12,2)
9. )
10. BEGIN
11.     INSERT INTO Employee_Type VALUES (employee_type_id, employee_type,
12.                                         employee_type_desc, pay_type, compensation_package);
12. END $$ 
13. DELIMITER ;

```

Result Grid | Filter Rows:  | Export: Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	employee_type_id	int(11)	NO	PRI	NULL	auto_increment
	employee_type	varchar(50)	YES		NULL	
	employee_type_desc	tinytext	YES		NULL	
	pay_type	varchar(10)	YES		NULL	
	compensation_package	decimal(12,2)	YES		NULL	

```

1. -- Stored procedure for inserting data into table Organisation --
2. DELIMITER $$ 
3. CREATE PROCEDURE insert_organization(
4. IN organization_id INT,
5. IN client_org_name VARCHAR(20),
6. IN client_org_code INT,
7. IN client_org_domain VARCHAR(20),
8. IN availability_date DATE,
9. IN top_level_name VARCHAR(20),
10. IN ISO_country_code VARCHAR(20)
11. )
12. BEGIN
13.     INSERT INTO organization VALUES (organization_id, client_org_name,
    client_org_code, client_org_domain, availability_date, top_level_name,
    ISO_country_code);
14. END $$ 
15. DELIMITER ;
16.

```

	organization_id	client_org_name	client_org_code	Client_org_domain	availability_date	top_level_name	ISO_country_code
▶	1	Boeing	3	Food	2020-07-12	level4	CN
	2	McD	4	Manufacturing	2021-02-15	level2	UA
	3	McD	3	Retail	2021-05-22	level2	BR
	4	Macy's	3	Banking	2019-06-13	level1	PT
	5	Boeing	2	Banking	2017-10-21	level3	FR
	6	Chase	4	Retail	2021-03-26	level4	DO
	7	Boeing	2	Banking	2017-08-28	level3	BR
	8	Chase	2	Retail	2021-12-21	level2	PH
	9	McD	2	Manufacturing	2020-03-01	level2	CN
*	10	Chase	3	Banking	2021-08-05	level1	CN
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```

1. -- Stored procedure for inserting data into table Person --
2. DELIMITER $$ 
3. CREATE PROCEDURE insert_person(
4. IN person_id INT,
5. IN first_name VARCHAR(20),
6. IN middle_name VARCHAR(20),
7. IN last_name VARCHAR(20),
8. IN age INT,
9. IN phone_number VARCHAR(15),
10. IN email_id VARCHAR(100),
11. IN address_id INT,
12. IN insurance_id VARCHAR(20),
13. IN device_type VARCHAR(20)
14. )
15. BEGIN
16.     INSERT INTO Person VALUES (person_id, first_name, middle_name, last_name, age,
    phone_number, email_id, address_id, insurance_id, device_type);
17. END $$ 
18. DELIMITER ;

```

	person_id	first_name	middle_name	last_name	age	phone_number	email_id		address_id	insurance_id	device_type
▶	1	Bessy	Stillman	Maxfield	53	110 205 3647	smaxfield0@sohu.com	4	U845AU475	iPad	
	2	Kinnee	Clemmie	Domsalla	21	447 716 7270	cdomsalla1@google.com	2	Q417IU671	Laptop	
	3	Idalina	Simone	Kopke	36	960 203 5645	skopke2@nifty.com	4	D100SP630	iPad	
	4	Serene	Bryanty	Vowels	42	604 978 8061	bvowels3@clickbank.net	3	G636IZ552	Laptop	
	5	Keene	Barde	Careswell	30	639 532 9345	bcareswell4@pagesperso-orange.fr	2	Z529GH958	Phone	
	6	Darcee	Rocky	Scown	33	688 717 3188	rscown5@scribd.com	2	T740ZG676	Laptop	
	7	Vaughn	Eb	Manwaring	29	938 818 4917	emanwaring6@macromedia.com	4	A100PV898	iPad	
	8	Stormi	Chaddie	Snead	43	170 481 4756	csnead7@state.gov	4	H012RA495	Phone	
	9	Briana	Jeff	Jizhaki	43	488 973 4214	jizhaki8@cafepress.com	4	Y419NX164	Phone	
	10	Dyann	Rudie	Windress	40	730 266 0008	rwindress9@taobao.com	2	L939V1421	Laptop	
	11	Tuck	Hendrik	Förbear	45	224 546 7984	hforbear@printfriendly.com	3	Z455GZ824	Phone	
	12	Elissa	Curry	Attridge	53	924 582 8679	cattridgeb@cbsnews.com	2	S313RK833	Laptop	
	13	Arié	Regan	Wase	42	982 501 8321	rwasec@phppb.com	4	O020UE805	Desktop	
	14	Freddy	Townsend	Duffell	37	835 953 5548	tduffeld@yellowpages.com	2	Y899CY225	Phone	
	15	Marrissa	Raynard	Tremblett	53	524 640 7332	rtremblette@lulu.com	2	R174E2691	iPad	
	16	Katya	Brian	McMurray	43	351 524 3668	bmcmurrayaf@seesaa.net	4	B140KV007	Phone	
	17	Jody	Hubie	Chave	48	299 534 4095	hchaveg@macromedia.com	4	J290YH065	Laptop	
	18	Kathie	Harp	Worwood	40	261 873 1813	hworwoodh@unc.edu	3	F823ZG395	Desktop	
	19	Torry	Raphael	Kaming	46	525 890 3683	rkamini@lulu.com	3	J934CY885	Phone	

#### 4. Query to insert data into tables

The data has to be inserted in the tables such that the constraints are all satisfied. INFORMATION\_SCHEMA TABLE\_CONSTRAINTS Table explains all the constraints and **helps to organize the sequence in which data is inserted into the tables.**

Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL
Name	Schema	Table		Column	Referenced Sch...	Referenced Table	Referenced Col...
employee_ibfk_2	project_employee	employee		employee_role_id	project_employee	employee_role	employee_role_id
employee_ibfk_4	project_employee	employee		employee_type_id	project_employee	employee_type	employee_type_id
employee_ibfk_1	project_employee	employee		organization_id	project_employee	organization	organization_id
employee_ibfk_3	project_employee	employee		person_id	project_employee	person	person_id
Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL
Name	Schema	Table		Column	Referenced Sch...	Referenced Table	Referenced Col...
person_ibfk_1	project_employee	person		address_id	project_employee	address_type	address_id
Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL
Name	Schema	Table		Column	Referenced Sch...	Referenced Table	Referenced Col...
employee_ibfk_2	project_employee	employee		employee_role_id	project_employee	employee_role	employee_role_id
Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL
Name	Schema	Table		Column	Referenced Sch...	Referenced Table	Referenced Col...
employee_ibfk_4	project_employee	employee		employee_type_id	project_employee	employee_type	employee_type_id
Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL
Name	Schema	Table		Column	Referenced Sch...	Referenced Table	Referenced Col...
employee_ibfk_1	project_employee	employee		organization_id	project_employee	organization	organization_id

Info	Columns	Indexes	Triggers	Foreign keys	Partitions	Grants	DDL
Name	Schema	Table		Column	Referenced Sch...	Referenced Table	Referenced Col...
employee_ibfk_3	project_employee	employee		person_id	project_employee	person	person_id
person_ibfk_1	project_employee	person		address_id	project_employee	address_type	address_id

```

1. call insert_employee_role (1,'IT','Support Maintanance');
2. call insert_address_type (1,'Home','3234 Lacos Street');
3. call insert_person
('1','Jack','Taylor','Albert',34,'878756778','taylor1@gmail.com',1,1,'Desktop');
4. call insert_Employee_Type ('1','Part-time','Emp Desc','Hourly', 25.5);
5. call insert_organization (1,'Developer','PM',1,'DIR1','2012-01-
01','Jameson','Manager','3166-2');
6. call insert_employee(1,1,1,1,1,'USA','USA','M','1997-01-01','Single','American');
```

## Example of Insertion of data using a stored procedure

```

1. --- Stored Procedure ---
2. DELIMITER $$ 
3. CREATE PROCEDURE load_data()
4. BEGIN
5. call insert_employee_role (1,'IT','Support Maintanance');
6. call insert_address_type (1,'Home','3234 Lacos Street');
7. call insert_person
('1','Jack','Taylor','Albert',34,'878756778','taylor1@gmail.com',1,1,'Desktop');
8. call insert_Employee_Type ('1','Part-time','Emp Desc','Hourly', 25.5);
9. call insert_organization (1,'Developer','PM',1,'DIR1','2012-01-
01','Jameson','Manager','3166-2');
10. call insert_employee(1,1,1,1,1,'USA','USA','M','1997-01-
01','Single','American');
11. END $$ 
12. DELIMITER ;
```

```

1. - Call statement to insert Data --
2. call load_data();
```

### Tip:

Error: Cannot delete or update a parent row: a foreign key constraint fails (Data integrity is done manually)

Data integrity can be disabled:

- 1) Delete foreign key
- 2) Use delete operation in SQL
- 3) Add the foreign key back to schema

## Best practices

1. Used table aliases where SQL statement involve more than one table, which eased code maintainability and readability.
2. Identification of database table structure before implementation helps to find the constraint issues.
3. Using specific column names to pull data is better than using star in Select queries, it improves performance when requesting large number of rows.
4. Assigning names to SQL objects and variables are done in such a way that they are short and meaningful.
5. Use of passing parameters to stored procedure to increase reusability.

## Alternative methods

Following are the alternative methods in which the View can be implemented:

1. Using ‘where’ clause instead of using ‘joins’ on multiple tables
2. Denormalizing the table and using that table for creating the view. It is best for performance as it reduces the overhead produced while joining multiple tables.

## References:

[1]<https://dev.mysql.com/doc/refman/8.0/en/table-constraints-table.html>

[2]<http://www.mysqltutorial.org/>

[3] <https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-stored-procedures.html>