DBMS PROJECT

ON

COURIER MANAGEMENT SYSTEM

DESIGNED BY:-

JAYANTHI ANSHU (22CSB0A14)

MURTAZA KAIZAR KUSHALGADHWALA (22CSB0A17)

PROBLEM STATEMENT :

To design and implement a database to help courier service businesses manage their employees, packages and customers.

REQUIREMENTS :

The solution must be able to solve the following:

- Keep track of existing customers and accommodate new ones
- Keep track of all employees and their details
- Assess the statistics and performance of each branch
- Maintain details of which localities a certain branch delivers to
- Maintain a record of all packages and their current status(delivered, shipped etc.)
- Receive Feedback from users
- View important business logistics such as quarterly revenue, annual revenue, performance of employees etc.

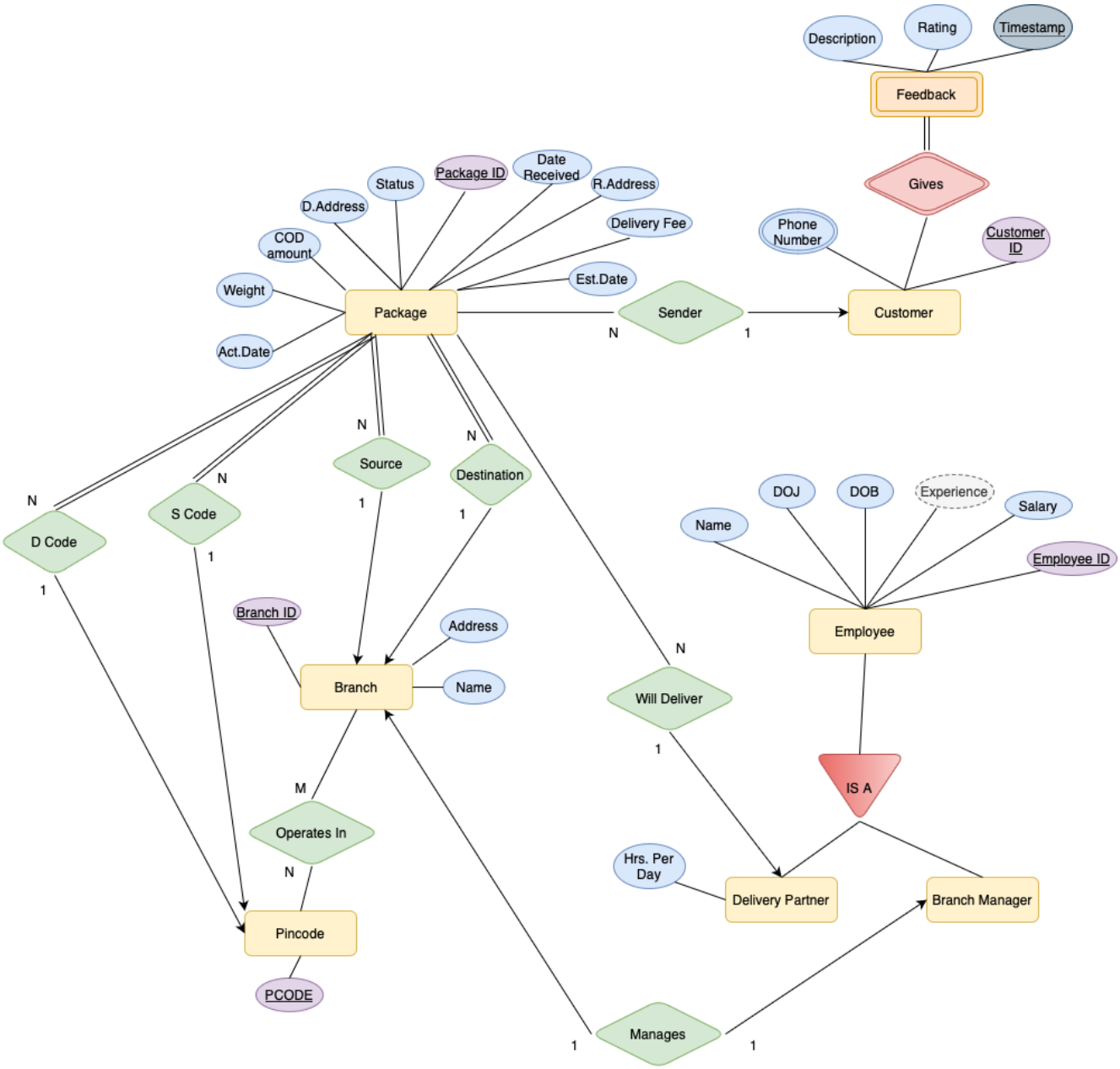
ASSUMPTIONS :

The following assumptions are being made:

- A single customer can send any number of packages at any given time
- An algorithm exists to attach a delivery fee to each package and allot delivery routes
- The sender will drop the parcel off at their nearest branch (Source Branch)
- A package will be sent from the source branch to a destination branch
- After a package is transported from the source branch to the destination brach, a delivery partner will be able to pick it up and deliver it to the receiver's doorstep.
- Each branch has a certain list of pincodes it operates in
- A package will be transferred to a destination branch which operates in the pincode of the receivers address
- Each branch has a unique branch manager
- A delivery partner is not associated with any specific branch and can deliver packages from any branch.
- A city and street name uniquely determine a pincode

SOLUTION

ER DIAGRAM:



NORMALISATION & CREATING TABLES :

1) ENTITY :- PACKAGE

Note) In the ER diagram, D.Address and R.Address are composite values. This is not accordance with 1NF form. Hence we decompose to street, city ,state and pincode.

-> as all values are now atomic the table is in 1NF

ATTRIBUTES:

- packageID (pk)
- Weight
- dateReceived
- dStreet
- dCity
- dState
- dPincode
- rStreet
- rCity
- rState
- rPincode
- CODamount
- deliveryFees
- estDelivery
- actualDelivery
- tstatus
- sBranch
- dBranch
- CustomerID
- empID

Prime Attributes :

packageID

Non Prime Attributes :

Remaining attributes are non prime

FUNCTIONAL DEPENDENCIES :

packageID -> all attributes (packageID is primary key)

dStreet,dCity -> dPincode (non prime -> non prime hence transitive dependency)

dPincode -> dCity (transitive dependency)

dCity->dState (transitive dependency)

rStreet,rCity -> rPincode (transitive dependency)

rPincode -> rCity (transitive dependency)

rCity->rState (transitive dependency)

NORMALISATION :

- As the table is in 1NF form and there are no partial dependencies (proper subset of key -> non prime) the table is in 2NF
- To convert it into 3NF form we must remove transitive dependencies.

We decompose the Package relation into 3 separate relations, they are:

1) Package(packageID, weight, dateReceived, dStreet, dPincode, rStreet, rPincode, CODamount, deliveryFees, estDelivery, actualDelivery, tstatus, sBranch, dBranch, CustomerID,empID)

2) Pincode(pcode, city)

3) Cities (city, State)

-> As all F.Ds in all 3 tables are of the form X->Y where X is a candidate key of the relation, Hence all 3 tables are in BCNF.

Constraints:

1) In Package :

packageID -> primary key
dPincode -> foreign key (refers pcode of Pincode)
rPincode -> foreign key (refers pcode of Pincode)
sBranch -> foreign key (refers branchID of Branch table)
dBranch -> foreign key (refers branchID of Branch table)
CustomerID -> foreign key(refers customerID of Customer table)
empID -> foreign key(refers empID of Employee table)

2) in Cities :

city -> primary key

Proof of Lossless Join:

1) Between Package and Pincode :

Package ∩ Pincode = pcode which is a primary key in Pincode table,Hence it is a lossless join

2) Between Pincode and City :

Pincode ∩ City = city which is a primary key in City table, Hence it is a lossless join.

Hence all three tables can be joined together without data loss, hence this is a lossless decomposition.

Proof of FD Preservation :

FDs of Package relation are :
packageID -> all attributes

FDs of Pincode relation are :
pcode -> city

FDs of Cities are :
city -> state

It is clear that FD(package) ∪ FD(Pincode) ∪ FD(City) = original FDs
Hence functional dependencies are preserved.

2) ENTITY :- BRANCH

Note) In the ER diagram, Address is a composite value. This is not in accordance with 1NF form. Hence we decompose to street, city ,state and pincode.

-> as all values are now atomic the table is in 1NF

ATTRIBUTES :

- BranchID (pk)
- Name

- Street
- City
- State
- Pincode

Prime Attributes :
BranchID

Non Prime Attributes :
Remaining attributes are non prime

FUNCTIONAL DEPENDENCIES :

BranchID -> all attributes(BranchID is primary key)

Street,City -> Pincode (non prime -> non prime hence transitive dependency)

Pincode -> City (transitive dependency)

City->State (transitive dependency)

NORMALISATION :

- As the table is in 1NF form and there are no partial dependencies (proper subset of key -> non prime) the table is in 2NF

- To convert it into 3NF form we must remove transitive dependencies.

We decompose the Package relation into 3 separate relations, they are:

1) Branch(BranchID,Name,Street,Pincode)

2) Pincode(pcode, city)

3) Cities (city, State)

-> As all F.Ds in all 3 tables are of the form X->Y where X is a candidate key of the relation, Hence all 3 tables are in BCNF.

Constraints:

1) In Branch :

BranchID-> primary key
pincode -> foreign key (refers pcode of Pincode)

Note 1) Pincode and Cities tables already exist in our schema

Note 2) proof of lossless join and preservation of FDs is similar to that above and has been omitted.

3) ENTITY :- CUSTOMER

ATTRIBUTES :

- CustomerID(pk)
- Ph1 (phone number)
- Ph2 (alternate number)
- fname
- lname

Prime Attributes :
CustomerID,Ph1,fname,lname

Non Prime Attributes :
Ph2

FUNCTIONAL DEPENDENCIES :

CustomerID -> all attributes(Customer ID is primary key)
Ph1,fname,lname -> CustomerID (prime -> prime so no issue wrt to 3NF)

NORMALISATION :

- As all F.Ds in the table are of the form X->Y where X is a candidate key of the relation, Hence the given table is in BCNF.

Constraints:

1) In Customer :
   CustomerID -> primary key
   Ph1,fname,lname -> primary key

4) ENTITY :- FEEDBACK

Note) as Feedback is a weak entity set, it will require the primary key of the strong entity set on which it is dependant (in this case Customer) combined with its discriminator (in this case ftime) to form a primary key for the weak entity set.

ATTRIBUTES :

- ftime (discriminator) (timestamp of the review)
- custid(part of primary key) (customer id)
- fdesc (description of the review)
- rating

Prime Attributes :
custid,ftime

Non Prime Attributes :
fdesc,rating

FUNCTIONAL DEPENDENCIES :

custid,ftime -> fdesc,rating

NORMALISATION :

- As all F.Ds in the table are of the form X->Y where X is a candidate key of the relation, Hence the given table is in BCNF.

Constraints:

1) In Feedback :
   custid,ftime-> primary key
   custid-> foreign key (refers CustomerID of Customer)

5) ENTITY :- OperatesIn

Used to determine which branches service which pincodes

ATTRIBUTES :

- BranchID
- pincode

Prime Attributes :
BranchID,pincode

Non Prime Attributes :
none

FUNCTIONAL DEPENDENCIES :

- BranchID,pincode -> BranchID,pincode

NORMALISATION :

- As all F.Ds in the table are of the form X->Y where X is a key of the relation, Hence the given table is in BCNF.

Constraints:

1) In OperatesIn :
   branchID,pincode-> primary key
   branchID-> foreign key (refers BranchID of Branch)
   pincode -> foreign key (refers pcode of Pincode)

6) ENTITY :- Employee

ATTRIBUTES :

- EmpID
- fname
- lname

- DOB
- DOJ
- Salary

Prime Attributes :
EmpID

Non Prime Attributes :
Remaining attributes of Employee

FUNCTIONAL DEPENDENCIES :

- EmpID -> fname, lname, DOB,DOJ,Salary

NORMALISATION :

- As all F.Ds in the table are of the form X->Y where X is a key of the relation, Hence the given table is in BCNF.

Constraints:

1) In Employee :
   EmpID-> primary key

7) ENTITY :- BranchManager

ATTRIBUTES :

- EmpID
- BranchID

Prime Attributes :
EmpID

Non Prime Attributes :
BranchID

FUNCTIONAL DEPENDENCIES :

- EmpID -> BranchID

NORMALISATION :

- As all F.Ds in the table are of the form X->Y where X is a key of the relation, Hence the given table is in BCNF.

Constraints:

1) In BranchManager :

   EmpID-> primary key
   EmpID -> foreign key(refers EmpID from Employee)
   BranchID -> foreign key(refers BranchID of Branch)

8) ENTITY :- DeliveryPartner

ATTRIBUTES :

- <u>EmpID</u>
- Hours

Prime Attributes :

EmpID

Non Prime Attributes :

Hours

FUNCTIONAL DEPENDENCIES :

- EmpID->Hours

NORMALISATION :

- As all F.Ds in the table are of the form X->Y where X is a key of the relation, Hence the given table is in BCNF.

Constraints:

1) In DeliveryPartner :

EmpID-> primary key
EmpID -> foreign key(refers EmpID from Employee)

9) ENTITY :- Pincode

ATTRIBUTES :

- pcode (pk)
- city

Prime Attributes :
pcode

Non Prime Attributes :
Hours

FUNCTIONAL DEPENDENCIES :

- pcode -> city

NORMALISATION :

- As all F.Ds in the table are of the form X->Y where X is a key of the relation, Hence the given table is in BCNF.

Constraints:

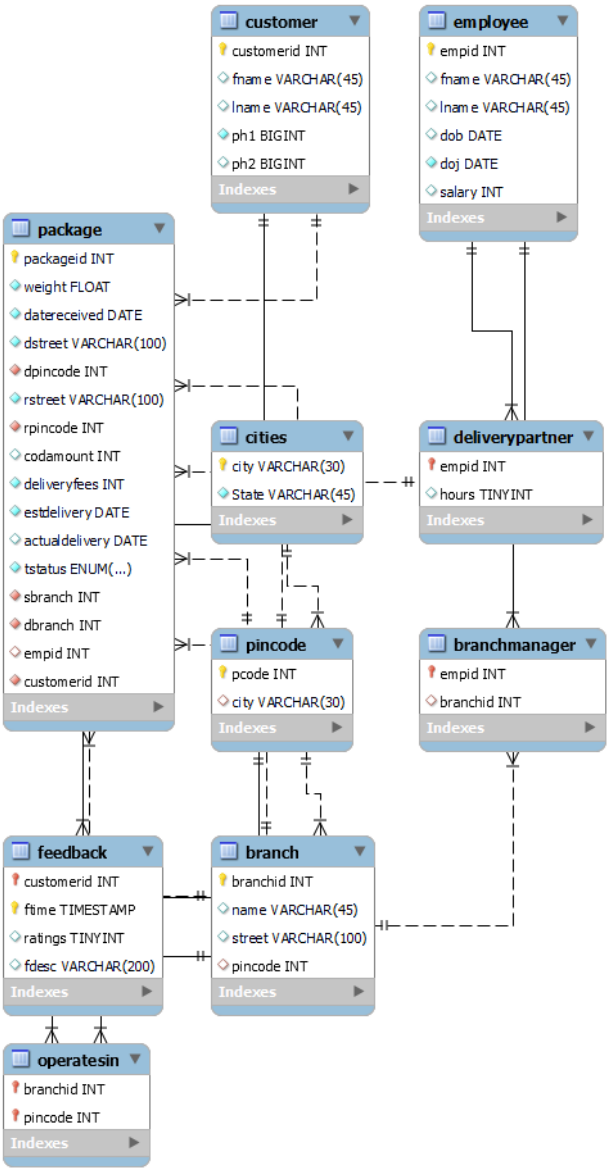
1) In Pincode :

pcode -> primary key
city -> foreign key(refers city of Cities table)

FINAL RELATIONS :

1. Package(PackageID,Weight, dateReceived, dStreet, dPincode, rStreet, rPincode, CODAmount, deliveryFees, estDelivery, actualDelivery, tstatus, sBranch, dBranch, CustomerID,empID)
2. Pincode(pcode,city)
3. Branch(BranchID, Name, street,pincode)
4. Customer(CustomerID,Ph1,Ph2)
5. Feedback(custid,fdesc,rating,ftime)
6. OperatesIn(BranchID,pincode)
7. Employee(EmpID,fname, lname, DOB,DOJ,Salary)
8. BranchManager (EmpID,BranchID)
9. DeliveryPartner(EmpID,Hours)
10. Cities(city,state)

The Schema can be shown Diagrammatically as below:



SQL TO CREATE TABLES :

```
CREATE SCHEMA `courier_service` ;
ENGINE = InnoDB;
```

1) Cities

```
 CREATE TABLE `courier_service`.`cities` (
`city` VARCHAR(30) NOT NULL,
`State` VARCHAR(45) NOT NULL,
 PRIMARY KEY (`city`));
```

2) Pincode

```
 CREATE TABLE `courier_service`.`pincode` (
`pcode` INT NOT NULL,
`city` VARCHAR(30) NULL,
 PRIMARY KEY (`pcode`),
 INDEX `city_idx` (`city` ASC) VISIBLE,
 CONSTRAINT `city`
 FOREIGN KEY (`city`)
 REFERENCES `courier_service`.`cities` (`city`)
 ON DELETE SET NULL
 ON UPDATE CASCADE);
```

3) Employee

```
 CREATE TABLE `courier_service`.`employee` (
`empid` INT NOT NULL AUTO_INCREMENT,
`fname` VARCHAR(45) NULL,
`lname` VARCHAR(45) NULL,
`dob` DATE NULL,
`doj` DATE NOT NULL,
`salary` INT NULL,
 PRIMARY KEY (`empid`));
```

4) Branch

```
 CREATE TABLE `courier_service`.`branch` (
`branchid` INT NOT NULL AUTO_INCREMENT,
`name` VARCHAR(45) NULL,
`street` VARCHAR(100) NULL,
`pincode` INT NULL,
 PRIMARY KEY (`branchid`),
 INDEX `pincode_idx` (`pincode` ASC) VISIBLE,
 CONSTRAINT `pincode`
 FOREIGN KEY (`pincode`)
 REFERENCES `courier_service`.`pincode` (`pcode`)
 ON DELETE SET NULL
 ON UPDATE CASCADE);
```

5) BranchManager

```
 CREATE TABLE `courier_service`.`branchmanager` (
`empid` INT NOT NULL,
`branchid` INT NULL,
 PRIMARY KEY (`empid`),
 UNIQUE INDEX `branchid_UNIQUE` (`branchid` ASC) VISIBLE,
 CONSTRAINT `empid`
 FOREIGN KEY (`empid`)
 REFERENCES `courier_service`.`employee` (`empid`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION,
 CONSTRAINT `branchid`
 FOREIGN KEY (`branchid`)
 REFERENCES `courier_service`.`branch` (`branchid`)
 ON DELETE SET NULL
 ON UPDATE CASCADE);
```

6) deliveryPartner

```
 CREATE TABLE `courier_service`.`deliverypartner` (
`empid` INT NOT NULL,
`hours` TINYINT NULL,
 PRIMARY KEY (`empid`),
 CONSTRAINT `empidfk`
 FOREIGN KEY (`empid`)
 REFERENCES `courier_service`.`employee` (`empid`)
 ON DELETE CASCADE
```

7) operatesIn

```
CREATE TABLE `courier_service`.`operatesin` (
`branchid` INT NOT NULL,
`pincode` INT NOT NULL,
 PRIMARY KEY (`branchid`, `pincode`),
 INDEX `pincodefk_idx` (`pincode` ASC) VISIBLE,
 CONSTRAINT `branchidfk`
 FOREIGN KEY (`branchid`)
 REFERENCES `courier_service`.`branch` (`branchid`)
 ON DELETE CASCADE
 ON UPDATE CASCADE,
 CONSTRAINT `pincodefk`
 FOREIGN KEY (`pincode`)
 REFERENCES `courier_service`.`pincode` (`pcode`)
 ON DELETE CASCADE
 ON UPDATE CASCADE);
```

8) Customer

```
CREATE TABLE `courier_service`.`customer` (
```

```
  `customerid` INT NOT NULL,
  `fname` VARCHAR(45) NULL,
  `lname` VARCHAR(45) NULL,
  `ph1` BIGINT NOT NULL,
  `ph2` BIGINT NULL,
   PRIMARY KEY (`customerid`));
```

9) Package

```
   CREATE TABLE `courier_service`.`package` (
  `packageid` INT NOT NULL AUTO_INCREMENT,
  `weight` FLOAT NOT NULL,
  `datereceived` DATE NOT NULL,
  `dstreet` VARCHAR(100) NOT NULL,
  `dpincode` INT NOT NULL,
  `rstreet` VARCHAR(100) NOT NULL,
  `rpincode` INT NOT NULL,
  `codamount` INT NULL,
  `deliveryfees` INT NOT NULL,
  `estdelivery` DATE NOT NULL,
  `actualdelivery` DATE NULL,
  `tstatus` ENUM('PROCESSING', 'DELIVERED', 'IN TRANSIT', 'OUT FOR
DELIVERY') NOT NULL,
  `sbranch` INT NOT NULL,
  `dbranch` INT NOT NULL,
  `empid` INT NULL,
  `customerid` INT NOT NULL,
   PRIMARY KEY (`packageid`),
   INDEX `sbranch_idx` (`sbranch` ASC) VISIBLE,
   INDEX `customers_idx` (`customerid` ASC) VISIBLE,
   INDEX `deliveryman_idx` (`empid` ASC) VISIBLE,
   INDEX `pincodefk_idx` (`dpincode` ASC) VISIBLE,
   INDEX `rpincode_idx` (`rpincode` ASC) VISIBLE,
   INDEX `dbranch_idx` (`dbranch` ASC) VISIBLE,
   CONSTRAINT `sbranch`
   FOREIGN KEY (`sbranch`)
   REFERENCES `courier_service`.`branch` (`branchid`)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT `customers`
   FOREIGN KEY (`customerid`)
   REFERENCES `courier_service`.`customer` (`customerid`)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT `deliveryman`
   FOREIGN KEY (`empid`)
   REFERENCES `courier_service`.`deliverypartner` (`empid`)
   ON DELETE SET NULL
   ON UPDATE CASCADE,
   CONSTRAINT `dpincode`
   FOREIGN KEY (`dpincode`)
```

```
  REFERENCES `courier_service`.`pincode` (`pcode`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT `rpincode`
  FOREIGN KEY (`rpincode`)
  REFERENCES `courier_service`.`pincode` (`pcode`)
  ON DELETE CASCADE
   ON UPDATE CASCADE,
   CONSTRAINT `dbranch`
   FOREIGN KEY (`dbranch`)
   REFERENCES `courier_service`.`branch` (`branchid`)
   ON DELETE CASCADE
   ON UPDATE CASCADE);
```

10) feedback

```
   CREATE TABLE `courier_service`.`feedback` (
  `customerid` INT NOT NULL,
  `ftime` TIMESTAMP NOT NULL,
  `ratings` TINYINT NULL,
  `fdesc` VARCHAR(200) NULL,
   PRIMARY KEY (`customerid`, `ftime`),
   CONSTRAINT `customerid`
   FOREIGN KEY (`customerid`)
   REFERENCES `courier_service`.`customer` (`customerid`)
   ON DELETE CASCADE
   ON UPDATE CASCADE);
```

SQL TO INSERT VALUES :

1)Cities

```
 INSERT INTO courier_service.cities (city, State) VALUES
('Mumbai', 'Maharashtra'),
('Delhi', 'Delhi'),
('Bangalore', 'Karnataka'),
('Chennai', 'Tamil Nadu'),
```

```
('Kolkata', 'West Bengal'),
('Hyderabad', 'Telangana');
```

	city	State
1	Bangalore	Karnataka
2	Chennai	Tamil Nadu
3	Delhi	Delhi
4	Hyderabad	Telangana
5	Kolkata	West Bengal
6	Mumbai	Maharashtra

2)Pincode

```
INSERT INTO courier_service.pincode (pcode, city) VALUES
(400001, 'Mumbai'),
(110001, 'Delhi'),
(560001, 'Bangalore'),
(600001, 'Chennai'),
(700001, 'Kolkata'),
(500001, 'Hyderabad'),
(500070, 'Hyderabad'),
(400050, 'Mumbai'),
(400075, 'Mumbai');
```

	pcode	city
1	560,001	Bangalore
2	600,001	Chennai
3	110,001	Delhi
4	500,001	Hyderabad
5	500,070	Hyderabad
6	700,001	Kolkata
7	400,001	Mumbai
8	400,050	Mumbai
9	400,075	Mumbai

3)Employee

```
INSERT INTO courier_service.employee (fname, lname, dob, doj,
salary) VALUES
('Ramesh', 'Kumar', '1990-05-15', '2015-07-10', 50000),
('Suresh', 'Verma', '1985-02-20', '2010-04-12', 55000),
('Priya', 'Gupta', '1992-08-12', '2016-09-20', 48000),
('Amit', 'Singh', '1988-04-25', '2014-03-15', 52000),
('Anita', 'Das', '1995-01-10', '2019-06-05', 49000),
('Rahul', 'Sharma', '1993-06-10', '2016-08-15', 45000),
('Amit', 'Gupta', '1990-07-20', '2017-09-12', 46000),
('Sneha', 'Verma', '1994-02-15', '2018-05-20', 44000),
('Vikas', 'Singh', '1988-11-25', '2015-04-10', 47000),
('Pooja', 'Das', '1991-04-05', '2019-06-25', 43000),
('Kiran', 'Reddy', '1987-09-20', '2014-11-10', 48000),
( 'Rohan', 'Kumar', '1985-05-15', '2010-08-20', 55000),
( 'Amit', 'Verma', '1989-07-12', '2015-06-10', 52000),
( 'Pooja', 'Sharma', '1992-03-25', '2018-09-05', 50000);
```

Grid Text

	empid	fname	lname	dob	doj	salary
1	1	Ramesh	Kumar	1990-05-15	2015-07-10	50,000
2	2	Suresh	Verma	1985-02-20	2010-04-12	55,000
3	3	Priya	Gupta	1992-08-12	2016-09-20	48,000
4	4	Amit	Singh	1988-04-25	2014-03-15	52,000
5	5	Anita	Das	1995-01-10	2019-06-05	49,000
6	6	Rahul	Sharma	1993-06-10	2016-08-15	45,000
7	7	Amit	Gupta	1990-07-20	2017-09-12	46,000
8	8	Sneha	Verma	1994-02-15	2018-05-20	44,000
9	9	Vikas	Singh	1988-11-25	2015-04-10	47,000
10	10	Pooja	Das	1991-04-05	2019-06-25	43,000
11	11	Kiran	Reddy	1987-09-20	2014-11-10	48,000
12	12	Rohan	Kumar	1985-05-15	2010-08-20	55,000
13	13	Amit	Verma	1989-07-12	2015-06-10	52,000
14	14	Pooja	Sharma	1992-03-25	2018-09-05	50,000

4)Branch

```
INSERT INTO courier_service.branch (name, street, pincode) VALUES
('Mumbai Central', 'ABC Street', 400001),
('Delhi North', 'XYZ Street', 110001),
('Bangalore East', 'PQR Street', 560001),
('Chennai West', 'LMN Street', 600001),
('Kolkata South', 'OPQ Street', 700001),
('Hyderabad North', 'JKL Street', 500001),
('Hyderabad East', 'XYZ Street', 500070),
('Mumbai West', 'LMN Street', 400001),
('Mumbai South', 'XYZ Street', 400050);
```

	branchid	name	street	pincode
1	1	Mumbai Central	ABC Street	400,001
2	2	Delhi North	XYZ Street	110,001
3	3	Bangalore East	PQR Street	560,001
4	4	Chennai West	LMN Street	600,001
5	5	Kolkata South	OPQ Street	700,001
6	6	Hyderabad North	JKL Street	500,001
7	7	Hyderabad East	XYZ Street	500,070
8	8	Mumbai West	LMN Street	400,001
9	9	Mumbai South	XYZ Street	400,050

5) branchManager

```
INSERT INTO courier_service.branchmanager (empid, branchid) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(11, 6),
(12, 7),
(13, 8),
(14, 9);
```

	empid	branchid
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	11	6
7	12	7
8	13	8
9	14	9

6) deliveryPartner

```
INSERT INTO courier_service.deliverypartner (empid, hours) VALUES
(6, 8),
(7, 7),
(8, 8),
(9, 7),
(10, 8);
```

	empid	branchid
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	11	6
7	12	7
8	13	8
9	14	9

7) operatesIn

```
INSERT INTO courier_service.operatesin (branchid, pincode) VALUES
(1, 400001),
(2, 110001),
(3, 560001),
(4, 600001),
(5, 700001),
(7, 500070),
(7, 500001),
(6, 500001),
(8, 400001),
(8, 400050),
(8, 400075),
(9, 400050),
(9, 400075);
```

	branchid	pincode
1	2	110,001
2	1	400,001
3	8	400,001
4	8	400,050
5	9	400,050
6	8	400,075
7	9	400,075
8	6	500,001
9	7	500,001
10	7	500,070
11	3	560,001
12	4	600,001
13	5	700,001

8)customer

```
INSERT INTO courier_service.customer (customerid, fname, lname,
ph1, ph2) VALUES
(1, 'Rohan', 'Kumar', 999990001, 999990011),
(2, 'Amit', 'Verma', 999990002, 999990012),
(3, 'Pooja', 'Sharma', 999990003, 999990013),
```

```
(4, 'Suresh', 'Menon', 999990004, 999990014),
(5, 'Priya', 'Nair', 999990005, 999990015),
(6, 'Ramesh', 'Iyer', 999990006, 999990016),
(7, 'Meena', 'Patel', 999990007, 999990017),
(8, 'Karthik', 'Raj', 999990008, 999990018),
(9, 'Neha', 'Singh', 999990009, 999990019),
(10, 'Anjali', 'Dutta', 999990010, 999990020);
```

	customerid	fname	lname	ph1	ph2
1	1	Rohan	Kumar	999,990,001	999,990,011
2	2	Amit	Verma	999,990,002	999,990,012
3	3	Pooja	Sharma	999,990,003	999,990,013
4	4	Suresh	Menon	999,990,004	999,990,014
5	5	Priya	Nair	999,990,005	999,990,015
6	6	Ramesh	Iyer	999,990,006	999,990,016
7	7	Meena	Patel	999,990,007	999,990,017
8	8	Karthik	Raj	999,990,008	999,990,018
9	9	Neha	Singh	999,990,009	999,990,019
10	10	Anjali	Dutta	999,990,010	999,990,020

9)feedback

```
INSERT INTO courier_service.feedback (customerid, ftime, ratings,
fdesc) VALUES
(1, '2024-03-27 10:15:00', 4, 'Good service'),
(2, '2024-03-27 11:20:00', 5, 'Excellent delivery'),
(3, '2024-03-28 09:45:00', 3, 'Late delivery but good
communication'),
(4, '2024-03-29 14:30:00', 4, 'Fast delivery and polite delivery
person'),
(5, '2024-03-30 16:00:00', 2, 'Package damaged upon arrival'),
(1, '2024-03-31 12:10:00', 5, 'Very satisfied with the service'),
(2, '2024-04-01 10:05:00', 4, 'Good tracking system and on-time
delivery'),
(3, '2024-04-02 11:50:00', 3, 'Average service'),
(4, '2024-04-03 13:20:00', 5, 'Excellent packaging and no
issues'),
(5, '2024-04-04 15:30:00', 1, 'Lost my package, very
disappointed');
```

	customerid	ftime	ratings	fdesc
1	1	2024-03-27 10:15:00	4	Good service
2	1	2024-03-31 12:10:00	5	Very satisfied with the service
3	2	2024-03-27 11:20:00	5	Excellent delivery
4	2	2024-04-01 10:05:00	4	Good tracking system and on-time delivery
5	3	2024-03-28 09:45:00	3	Late delivery but good communication
6	3	2024-04-02 11:50:00	3	Average service
7	4	2024-03-29 14:30:00	4	Fast delivery and polite delivery person
8	4	2024-04-03 13:20:00	5	Excellent packaging and no issues
9	5	2024-03-30 16:00:00	2	Package damaged upon arrival
10	5	2024-04-04 15:30:00	1	Lost my package, very disappointed

10) Package

```
INSERT INTO courier_service.package
(weight, datereceived, dstreet, dpincode, rstreet, rpincode,
codamount, deliveryfees, estdelivery, actualdelivery, tstatus,
sbranch, dbranch, empid, customerid)
values

(2.3, '2024-05-11', 'LMN Street', 600001, 'XYZ Street', 110001,
2100, 150, '2024-05-15', NULL, 'DELIVERED', 4, 2, NULL, 6),
(1.7, '2024-05-12', 'JKL Street', 400050, 'RST Street', 500001,
1400, 100, '2024-05-16', NULL, 'IN TRANSIT', 8, 5, NULL, 3),
(3.4, '2024-05-13', 'GHI Street', 400075, 'OPQ Street', 700001,
2900, 210, '2024-05-17', NULL, 'OUT FOR DELIVERY', 9, 4, NULL, 7),
(2.6, '2024-05-14', 'RST Street', 500001, 'UVW Street', 500070,
2300, 160, '2024-05-18', NULL, 'DELIVERED', 7, 3, NULL, 8),
(2.0, '2024-05-15', 'ABC Street', 400001, 'PQR Street', 560001,
1800, 130, '2024-05-19', NULL, 'IN TRANSIT', 6, 1, NULL, 4),
(2.8, '2024-05-16', 'PQR Street', 560001, 'LMN Street', 600001,
2500, 180, '2024-05-20', NULL, 'DELIVERED', 3, 7, NULL, 2),
(1.9, '2024-05-17', 'UVW Street', 500070, 'GHI Street', 400075,
1700, 120, '2024-05-21', NULL, 'OUT FOR DELIVERY', 7, 3, NULL, 1),
(3.1, '2024-05-18', 'OPQ Street', 700001, 'JKL Street', 400050,
2800, 190, '2024-05-22', NULL, 'DELIVERED', 8, 5, NULL, 9),
(2.4, '2024-05-19', 'LMN Street', 600001, 'RST Street', 500001,
2200, 160, '2024-05-23', NULL, 'IN TRANSIT', 4, 2, NULL, 5),
(2.2, '2024-05-20', 'XYZ Street', 110001, 'OPQ Street', 700001,
2000, 140, '2024-05-24', NULL, 'DELIVERED', 2, 4, NULL, 8),
(2.5, '2024-05-21', 'JKL Street', 400050, 'ABC Street', 400001,
2100, 150, '2024-05-25', NULL, 'OUT FOR DELIVERY', 6, 1, NULL, 3),
(2.3, '2024-05-22', 'GHI Street', 400075, 'UVW Street', 500070,
1900, 130, '2024-05-26', NULL, 'DELIVERED', 7, 3, NULL, 4),
(1.6, '2024-05-23', 'PQR Street', 560001, 'LMN Street', 600001,
1300, 90, '2024-05-27', NULL, 'IN TRANSIT', 3, 7, NULL, 5),
(3.0, '2024-05-24', 'RST Street', 500001, 'XYZ Street', 110001,
2600, 190, '2024-05-28', NULL, 'DELIVERED', 2, 4, NULL, 9),
(2.1, '2024-05-25', 'OPQ Street', 700001, 'JKL Street', 400050,
1800, 120, '2024-05-29', NULL, 'IN TRANSIT', 8, 5, NULL, 7),
(2.7, '2024-05-26', 'ABC Street', 400001, 'PQR Street', 560001,
2400, 170, '2024-05-30', NULL, 'DELIVERED', 6, 1, NULL, 2),
(2.9, '2024-05-27', 'LMN Street', 600001, 'GHI Street', 400075,
2700, 200, '2024-05-31', NULL, 'OUT FOR DELIVERY', 4, 2, NULL, 9),
(2.7, '2024-06-07', 'ABC Street', 400001, 'UVW Street', 500070,
2400, 170, '2024-06-11', NULL, 'DELIVERED', 6, 1, NULL, 2),
```

```
(1.6, '2024-06-08', 'LMN Street', 600001, 'PQR Street', 560001,
1500, 100, '2024-06-12', NULL, 'IN TRANSIT', 4, 2, NULL, 5),
(2.9, '2024-06-09', 'RST Street', 500001, 'GHI Street', 400075,
2600, 190, '2024-06-13', NULL, 'DELIVERED', 7, 3, NULL, 9),
(2.2, '2024-06-10', 'JKL Street', 400050, 'XYZ Street', 110001,
2100, 150, '2024-06-14', NULL, 'OUT FOR DELIVERY', 8, 5, NULL, 3),
(3.0, '2024-06-11', 'OPQ Street', 700001, 'LMN Street', 600001,
2700, 200, '2024-06-15', NULL, 'DELIVERED', 3, 7, NULL, 6),
(2.5, '2024-06-12', 'UVW Street', 500070, 'OPQ Street', 700001,
2200, 160, '2024-06-16', NULL, 'IN TRANSIT', 9, 4, NULL, 1),
(1.8, '2024-06-13', 'ABC Street', 400001, 'JKL Street', 400050,
1900, 130, '2024-06-17', NULL, 'DELIVERED', 6, 1, NULL, 8),
(2.4, '2024-06-14', 'PQR Street', 560001, 'RST Street', 500001,
2300, 160, '2024-06-18', NULL, 'OUT FOR DELIVERY', 5, 2, NULL, 7),
(2.7, '2024-06-15', 'XYZ Street', 110001, 'GHI Street', 400075,
2500, 180, '2024-06-19', NULL, 'DELIVERED', 2, 4, NULL, 9),
(1.9, '2024-06-16', 'LMN Street', 600001, 'UVW Street', 500070,
2000, 140, '2024-06-20', NULL, 'IN TRANSIT', 4, 3, NULL, 5),
(2.3, '2024-06-17', 'OPQ Street', 700001, 'ABC Street', 400001,
2100, 150, '2024-06-21', NULL, 'DELIVERED', 9, 1, NULL, 6),
(3.1, '2024-06-18', 'RST Street', 500001, 'JKL Street', 400050,
2700, 200, '2024-06-22', NULL, 'OUT FOR DELIVERY', 7, 5, NULL, 2),
(2.0, '2024-06-19', 'GHI Street', 400075, 'PQR Street', 560001,
1800, 120, '2024-06-23', NULL, 'DELIVERED', 1, 6, NULL, 3),
(2.8, '2024-06-20', 'UVW Street', 500070, 'LMN Street', 600001,
2600, 190, '2024-06-24', NULL, 'IN TRANSIT', 9, 4, NULL, 7),
(2.6, '2024-06-21', 'ABC Street', 400001, 'RST Street', 500001,
2400, 170, '2024-06-25', NULL, 'DELIVERED', 6, 1, NULL, 8),
(2.1, '2024-06-22', 'PQR Street', 560001, 'GHI Street', 400075,
2000, 140, '2024-06-26', NULL, 'OUT FOR DELIVERY', 5, 2, NULL, 9),
(2.9, '2024-06-23', 'JKL Street', 400050, 'XYZ Street', 110001,
2800, 210, '2024-06-27', NULL, 'DELIVERED', 8, 5, NULL, 3),
(2.4, '2024-06-24', 'OPQ Street', 700001, 'UVW Street', 500070,
2300, 160, '2024-06-28', NULL, 'IN TRANSIT', 7, 3, NULL, 4),
(2.5, '2024-06-25', 'LMN Street', 600001, 'ABC Street', 400001,
2200, 150, '2024-06-29', NULL, 'DELIVERED', 4, 2, NULL, 6),
(2.3, '2024-06-26', 'RST Street', 500001, 'PQR Street', 560001,
2100, 140, '2024-06-30', NULL, 'OUT FOR DELIVERY', 3, 7, NULL, 1);
```

	packageid	weight	datereceived	dstreet	dpincode	rstreet	rpincode	c
1	75	2.3	2024-05-11	LMN Street	600,001	XYZ Street	110,001	
2	76	1.7	2024-05-12	JKL Street	400,050	RST Street	500,001	
3	77	3.4	2024-05-13	GHI Street	400,075	OPQ Street	700,001	
4	78	2.6	2024-05-14	RST Street	500,001	UVW Street	500,070	
5	79	2	2024-05-15	ABC Street	400,001	PQR Street	560,001	
6	80	2.8	2024-05-16	PQR Street	560,001	LMN Street	600,001	
7	81	1.9	2024-05-17	UVW Street	500,070	GHI Street	400,075	
8	82	3.1	2024-05-18	OPQ Street	700,001	JKL Street	400,050	
9	83	2.4	2024-05-19	LMN Street	600,001	RST Street	500,001	
10	84	2.2	2024-05-20	XYZ Street	110,001	OPQ Street	700,001	
11	85	2.5	2024-05-21	JKL Street	400,050	ABC Street	400,001	
12	86	2.3	2024-05-22	GHI Street	400,075	UVW Street	500,070	
13	87	1.6	2024-05-23	PQR Street	560,001	LMN Street	600,001	
14	88	3	2024-05-24	RST Street	500,001	XYZ Street	110,001	
15	89	2.1	2024-05-25	OPQ Street	700,001	JKL Street	400,050	
16	90	2.7	2024-05-26	ABC Street	400,001	PQR Street	560,001	
17	91	2.9	2024-05-27	LMN Street	600,001	GHI Street	400,075	
18	92	2.7	2024-06-07	ABC Street	400,001	UVW Street	500,070	
19	93	1.6	2024-06-08	LMN Street	600,001	PQR Street	560,001	
20	94	2.9	2024-06-09	RST Street	500,001	GHI Street	400,075	
21	95	2.2	2024-06-10	JKL Street	400,050	XYZ Street	110,001	
22	96	3	2024-06-11	OPQ Street	700,001	LMN Street	600,001	
23	97	2.5	2024-06-12	UVW Street	500,070	OPQ Street	700,001	
24	98	1.8	2024-06-13	ABC Street	400,001	JKL Street	400,050	
25	99	2.4	2024-06-14	PQR Street	560,001	RST Street	500,001	
26	100	2.7	2024-06-15	XYZ Street	110,001	GHI Street	400,075	
27	101	1.9	2024-06-16	LMN Street	600,001	UVW Street	500,070	
28	102	2.3	2024-06-17	OPQ Street	700,001	ABC Street	400,001	
29	103	3.1	2024-06-18	RST Street	500,001	JKL Street	400,050	
30	104	2	2024-06-19	GHI Street	400,075	PQR Street	560,001	
31	105	2.8	2024-06-20	UVW Street	500,070	LMN Street	600,001	
32	106	2.6	2024-06-21	ABC Street	400,001	RST Street	500,001	
33	107	2.1	2024-06-22	PQR Street	560,001	GHI Street	400,075	
34	108	2.9	2024-06-23	JKL Street	400,050	XYZ Street	110,001	
35	109	2.4	2024-06-24	OPQ Street	700,001	UVW Street	500,070	
36	110	2.5	2024-06-25	LMN Street	600,001	ABC Street	400,001	
37	111	2.3	2024-06-26	RST Street	500,001	PQR Street	560,001	

TRIGGERS :

-> we employ triggers to help us automatically interact with our database in response to certain events

-> helps in maintaining database easily

1) OFD (Out for Delivery)

-> when a delivery partner is assigned to a package, the status of that package must automatically be change to ‘Out for delivery’.

-> This can be performed using a trigger

Code for Trigger :

```
delimiter //
        CREATE TRIGGER OFD BEFORE UPDATE ON package
        FOR EACH ROW
        BEGIN
        if new.empid <> old.empid then
        IF NEW.empid is not null THEN
        SET NEW.tstatus = 'OUT FOR DELIVERY';
        END IF;
         end if;
        END;//
delimiter ;
```

2) Delivered

-> when actual delivery is set it means the package has been delivered so the status of package must be automatically changed to ‘Delivered’

-> This can be done with the help of a trigger

Code for Trigger :

```
delimiter //
        CREATE TRIGGER delivered BEFORE UPDATE ON package
        FOR EACH ROW
        BEGIN
        if new.actualdelivery <> old.actualdelivery then
        IF NEW.actualdelivery is not null THEN
        SET NEW.tstatus = 'DELIVERED';
        END IF;
         end if;
        END;//
delimiter ;
```

IMPORTANT QUERIES :

1) Employee of the Month

-> we use a stored procedure to determine employee of the month for a given (month, year). Employee of the Month is the delivery partner who delivered most weight/hour in that given month.

Query :

```
delimiter //
create procedure getEmployeeOfTheMonth(in years int,in months int)
begin
select t1.empid, avg(t1.x) as workperheour
from (select package.empid,estdelivery,sum(weight)/hours as x
from package,deliverypartner
where package.empid=deliverypartner.empid and
month(estdelivery)=months and year(estdelivery)=years and tstatus
= 'DELIVERED'
group by empid,estdelivery) t1
group by empid order by avg(t1.x) desc limit 1;
end //
delimiter ;
```

Sample Output :

```
call getEmployeeOfTheMonth(2024,5);
```

	empid	workperheour
▶	7	0.42857142857142855

2) Branch of the Month

-> we use a stored procedure to determine branch of the month for a given (month,year). Branch of the Month is the branch which generated largest revenue that month.

-> Branch revenue is calculated as sum of delivery fees of all packages for which the branch is either a source or destination

Query :

```
delimiter //
create procedure getBranchOfTheMonth(in years int,in months int)
begin
select sbranch as branches ,sum(f) as fees_collected from
(select sbranch,sum(deliveryfees) as f from package where
month(estdelivery) =months and year(estdelivery)= years group by
sbranch
union
```

```
select dbranch,sum(deliveryfees) from package where
month(estdelivery) =months and year(estdelivery)= years group by
dbranch)
t1 group by sbranch order by sum(f) desc limit 1;
end //
delimiter ;
```

Sample output :

```
call getBranchOfTheMonth(2024,5);
```

	branches	fees_collected
▶	4	1050

3) Branch of the Quarter

->similar to Branch of the Month except we are returning branch of the month for a (year,quarter)

Query :

```
delimiter //
create procedure getBranchOfTheQuarter(in years int,in q int)
begin
select sbranch as branches ,sum(f) as fees_collected from
(select sbranch,sum(deliveryfees) as f from package where
month(estdelivery) in (3*q,3*q-2,3*q-1) and year(estdelivery)=
years group by sbranch
union
select dbranch,sum(deliveryfees) from package where
month(estdelivery) in (3*q,3*q-2,3*q-1) and year(estdelivery)=
years group by dbranch)
 t1 group by sbranch order by sum(f) desc limit 1;
end //
delimiter ;
```

Sample Output :

```
call getBranchOfTheQuarter(2024,2);
```

	branches	fees_collected
▶	4	1970

4) City wise Revenue

-> we use a stored procedure to determine the city with highest revenue in a given year.

-> The revenue of a city is defined as the sum of delivery fees of all packages for which the given city is the source city.

Query:

```
delimiter //
create procedure getCitywiseRevenue(in years int)
begin
select city,sum(deliveryfees) as Annual_Revenue
from package
join pincode
on dpincode=pcode
where year(estdelivery)=years
group by city order by sum(deliveryfees) desc;
end //
delimiter ;
```

Sample Output :

```
call getcitywiserevenue(2024);
```

	city	Annual_Revenue
▶	Mumbai	1840
	Hyderabad	1350
	Chennai	900
	Kolkata	820
	Bangalore	570
	Delhi	320

VIEWS :

-> Views have been implemented to facilitate easy viewing of important data.

-> Views also create abstraction and help improve data base security.

1) Processing Backlog

-> Here we create a view to display each branch along with how many packages are being currently processed in the branch.

View :

```
create view ProcessingBacklog as
select sbranch,count(tstatus='PROCESSING')
from package
group by sbranch
order by count(tstatus='PROCESSING') desc;
```

Sample output :

```
SELECT * FROM courier_service.processingbacklog;
```

sbranch	count(tstatus='PROCESSING')
4	6
6	6
7	6
8	5
3	4
9	4
2	3
5	2
1	1

2) Delivery Backlog

-> Here we create a view to display each branch along with how many packages are being currently out for delivery in the branch.

View :

```
create view DeliveryBacklog as
select empid,count(tstatus='OUT FOR DELIVERY')
from package
group by empid
order by count(tstatus='OUT FOR DELIVERY');
```

Sample output :

```
SELECT * FROM courier_service.deliverybacklog;
```

empid	count(tstatus='OUT FOR DELIVERY')
9	5
7	7
8	7
6	9
10	9

3) View Top Customers

-> Here we create a view to display information about customers who have generated high revenue for the service

-> Revenue of a customer is defined as the sum of delivery fees of all packages sent by that customer.

Query :

```
create view TopCustomers as
select package.customerid ,sum(deliveryfees),fname,lname,ph1,ph2
from package
join customer
on customer.customerid=package.customerid
group by package.customerid
order by sum(deliveryfees) desc;
```

Sample output :

```
SELECT * FROM courier_service.TopCustomers;
```

customerid	sum(deliveryfees)	fname	lname	ph1	ph2
9	1090	Neha	Singh	999990009	999990019
3	730	Pooja	Sharma	999990003	999990013
2	720	Amit	Verma	999990002	999990012
7	680	Meena	Patel	999990007	999990017
6	650	Ramesh	Iyer	999990006	999990016
8	600	Karthik	Raj	999990008	999990018
5	490	Priya	Nair	999990005	999990015
1	420	Rohan	Kumar	999990001	999990011
4	420	Suresh	Menon	999990004	999990014